robotron

RVS-Rechnersystem K 1840

Diagnosehandbuch Rechner

VEB Robotron-Vertrieb Berlin

RVS-Rechnersystem K 1840

Diagnosehandbuch Rechner

VEB Robotron-Vertrieb Berlin

Diese Unterlage ist Bestandteil der Gesamtdokumentation "RVS-Rechnersystem K 1840".

Die vorliegende Dokumentation entspricht dem Stand von 06/88.

Die Ausarbeitung erfolgte durch ein Kollektiv des VEB Robotron-Elektronik Dresden Stammbetrieb des VEB Kombinat Robotron.

Nachdruck, jegliche Vervielfältigung oder Auszüge daraus sind unzulässig.

Herausgeber:

VEB Robotron-Vertrieb Berlin
Mohrenstr. 62
Berlin 1086

Inhaltsverzeichnis Seite

1 ÜBERSICHT 1-1

1.1. DIAGNOSESYSTEMSTRUKTUR DES RVS K 1840 1-1
1.1.1. Allgemeines 1-1
1.1.2. Ebenen des Diagnosesystems 1-2
1.1.3. Struktur der Diagnose-Umgebung 1-3
1.1.4. Laden der Diagnoseprogramme und deren Steuerung (Übersicht) 1-6
1.1.5. Diagnosestrategie 1-8
1.1.6. Kodierungskonvention für Diagnoseprogramme 1-8
1.1.7. Die Diagnosedirectory SYSMAINT und deren Pflege 1-10

1.2. DATEIEN DER KONSOLEBENE UND DEREN AUFGABE 1-10
1.2.1. Übersicht über die Dateien und die zugeordneten Disketten 1-10
1.2.2. Diskette DS0 (Konsolsubsystemtest) 1-13
1.2.3. Diskette DS1 (RVS K 1840 Standardkonsole) 1-14
1.2.3.1. Aufgaben der Standardkonsoldiskette 1-14
1.2.3.2. Booten des Konsolprogramms 1-14
1.2.3.3. Booten des SVP 1800 von verschiedenen Datenträgern 1-15
1.2.3.4. Booten des Diagnose-Supervisors von verschiedenen Datenträgern (Stand-alone) 1-17
1.2.3.5. Restart des SVP 1800 vom Konsol-E/A-Modus 1-18
1.2.3.6. CRASH-Kommandodatei 1-19
1.2.4. Diskette DS2 (K 1840 MIKRODIA #1)
Diskette DS3 (K 1840 MIKRODIA #2)
Diskette DS4 (reserviert für Ergänzungen) 1-19

1.3. DATEIEN DER DIAGNOSEEBENE 4 1-19

1.4. DATEIEN DER DIAGNOSEEBENEN 3 UND 2 1-20
1.4.1. Übersicht 1-20
1.4.2. Disketten 1-20
1.4.3. Laden und Start der Diagnoseprogramme der Ebenen 2 und 3 im Stand-alone-Modus 1-21
1.4.3.1. Laden und Start des Supervisors von der Diskette 1-21
1.4.3.2. Booten des Supervisors von der Diskette 1-21
1.4.3.3. Booten des Supervisors vom Systemgerät 1-22
1.4.4. Start der Diagnoseprogramme der Ebene 2 im online-Modus 1-22

1.5. DATEIEN DER DIAGNOSEEBENE 2R 1-22

1.6. DATEIEN DER DIAGNOSEEBENE 1 1-22
1.6.1. Allgemeines 1-22
1.6.2. User Environment Test Package (UETP) 1-23
1.6.2.1. Einloggen und Vorbereitung der Geräte für den Test durch UETP 1-23
1.6.2.2. Start des UETP 1-24
1.6.2.3. Unterbrechung des UETP 1-24
1.6.2.4. Fehler 1-25
1.6.2.5. Tests 1-25
1.6.2.6. Abschluß des UETP 1-25
1.6.3. System Dump Analyzer (SDA) 1-26
1.6.3.1. Aufruf 1-26
1.6.3.2. ANALYZE/CRASH_DUMP 1-26
1.6.3.3. ANALYZE/SYSTEM 1-27
1.6.3.4. ANALYZE/ERROR_LOG 1-27

K O N S O L T E S T S 2-1

2.1. FEHLERORTUNG 2-1
2.1.1. Diagnosemöglichkeiten 2-1
2.1.2. Fehlersuchstrategie 2-1
2.1.2.1. Fehlersuche ZVE K 2662 2-2
2.1.2.2. Fehlersuche Speicher OPS K 3564.01 2-3
2.1.2.3. Fehlersuche für AIS K 8060.02 2-3
2.1.2.4. Fehlersuche für AFS K 5163 2-4
2.1.2.5. Fehlersuche für BAA 2-4
2.1.2.6. Fehlersuche mittels Bediengerät 2-4
2.1.2.7. Prüfung der Betriebsspannungen und der elektrischen Verbindungen 2-8

2.2. BESCHREIBUNG DES LEITPROGRAMMS LPROEM 2-8
2.2.1. Benennung 2-8
2.2.2. Dokumentationsnummer 2-8
2.2.3. Programmumfang 2-8
2.2.4. Speicherbedarf 2-9
2.2.5. Ausführungszeit 2-9
2.2.6. Testhilfsmittel/Geräteeinstellungen 2-9
2.2.7. Bedienungsanleitung für LPROEM 2-9
2.2.7.1. Aufruf des LPROEM 2-9
2.2.7.2. Steuerkennzeichen für Bedienereingaben 2-9
2.2.7.3. Aufruf LPRO-eigene Steuerroutinen 2-10
2.2.7.4. Steuerkommandos 2-11
2.2.7.5. Durch LPROEM erzeugte Fehlerausschriften 2-12
2.2.7.6. Nutzung der Bedieneinheit K 8911 2-14
2.2.8. Prüfprogrammbearbeitung über LPROEM 2-14
2.2.8.1. Anschlußbedingungen der Prüfprogramme 2-14
2.2.8.2. Anfangseinstellung des LPROEM 2-15
2.2.8.3. Durchführung Einzeltest 2-15
2.2.8.4. Durchführung Dauertest 2-15
2.2.8.5. Unterbrechungsbehandlung durch LPROEM 2-16

2.2.9. Fehlerprotokollierung 2-17
2.2.9.1. Aufbau der Fehlerausschriften 2-17
2.2.9.2. Bedeutung der Kurzzeichen in Fehlerausschriften 2-17
2.2.9.3. Fehlerformen 2-19
2.2.10. Anfangsladen des LPROEM und Bedienung der Foliienspeichereinheit des Konsolrechners 2-20
2.2.10.1. Start über Anfangslader 2-20
2.2.10.2. Gerätebedienung FSE 2-20
2.2.10.3. Programmtechnische Realisierung 2-21

2.3. BESCHREIBUNG DES PRÜFPROGRAMMS ZVZW20 2-21
2.3.1. Benennung 2-21
2.3.2. Dokumentationsnummer 2-21
2.3.3. Programmumfang 2-21
2.3.4. Speicherbedarf 2-22
2.3.5. Ausführungszeit 2-22
2.3.6. Testhilfsmittel/Geräteeinstellungen 2-22
2.3.7. Arbeit unter LPROEM 2-22
2.3.8. Arbeit ohne LPROEM 2-22
2.3.9. Generierinformationen 2-22
2.3.10. Fehlerinterpretation 2-23
2.3.10.1. Arbeitsweise mit LPROEM 2-23
2.3.10.2. Arbeitsweise ohne LPROEM 2-23
2.3.10.3. Funktionsbeschreibung 2-23

2.4. BESCHREIBUNG DES PRÜFPROGRAMMS ZARI20 2-24
2.4.1. Benennung 2-24
2.4.2. Dokumentationsnummer 2-24
2.4.3. Programmumfang 2-24
2.4.4. Speicherbedarf 2-25
2.4.5. Ausführungszeit 2-25
2.4.6. Arbeit unter LPROEM 2-25
2.4.7. Arbeit ohne LPROEM 2-25
2.4.8. Generierinformationen 2-25
2.4.9. Fehlerinterpretation 2-26
2.4.9.1. Arbeitsweise mit LPROEM 2-26
2.4.9.2. Arbeitsweise ohne LPROEM 2-26
2.4.10. Funktionsbeschreibung 2-26

2.5. BESCHREIBUNG DES PRÜFPROGRAMMS ZSHI20 2-27
2.5.1. Benennung 2-27
2.5.2. Dokumentationsnummer 2-27
2.5.3. Programmumfang 2-27
2.5.4. Speicherbedarf 2-27
2.5.5. Ausführungszeit 2-27
2.5.6. Testhilfsmittel/Geräteeinstellungen 2-27
2.5.7. Arbeit unter LPROEM 2-27
2.5.8. Arbeit ohne LPROEM 2-28
2.5.9. Generierinformationen 2-28
2.5.10. Fehlerinterpretation 2-28
2.5.10.1. Arbeitsweise mit LPROEM 2-28
2.5.10.2. Arbeitsweise ohne LPROEM 2-29

2.5.11. Funktionsbeschreibung 2-29

2.6. BESCHREIBUNG DES PRÜFPROGRAMMS ZBBI20 2-29
2.6.1. Benennung 2-29
2.6.2. Dokumentationsnummer 2-29
2.6.3. Programmumfang 2-29
2.6.4. Speicherbedarf 2-30
2.6.5. Ausführungszeit 2-30
2.6.6. Testhilfsmittel/Geräteeinstellungen 2-30
2.6.7. Arbeit unter LPROEM 2-30
2.6.8. Arbeit ohne LPROEM 2-30
2.6.9. Generierinformationen 2-31
2.6.10. Fehlerinterpretation 2-31
2.6.10.1. Arbeitsweise mit LPROEM 2-31
2.6.10.2. Arbeitsweise ohne LPROEM 2-31
2.6.11. Funktionsbeschreibung 2-32

2.7. BESCHREIBUNG DES PRÜFPROGRAMMS ZBEF30 2-32
2.7.1. Benennung 2-32
2.7.2. Dokumentationsnummer 2-32
2.7.3. Programmumfang 2-32
2.7.4. Speicherbedarf 2-33
2.7.5. Ausführungszeit 2-33
2.7.6. Testhilfsmittel/Geräteeinstellungen 2-33
2.7.7. Arbeit unter LPROEM 2-34
2.7.8. Arbeit ohne LPROEM 2-34
2.7.9. Generierinformationen 2-34
2.7.10. Fehlerinterpretation 2-34
2.7.10.1. Arbeitsweise mit LPROEM 2-34
2.7.10.2. Arbeitsweise ohne LPROEM 2-35
2.7.11. Funktionsbeschreibung 2-35

2.8. BESCHREIBUNG DES PRÜFPROGRAMMS ZADRAR 2-37
2.8.1. Benennung 2-37
2.8.2. Dokumentationsnummer 2-37
2.8.3. Programmumfang 2-37
2.8.4. Speicherbedarf 2-37
2.8.5. Ausführungszeit 2-37
2.8.6. Testhilfsmittel/Geräteeinstellungen 2-37
2.8.7. Arbeit unter LPROEM 2-37
2.8.8. Arbeit ohne LPROEM 2-38
2.8.9. Generierinformationen 2-38
2.8.10. Fehlerinterpretation 2-38
2.8.10.1. Arbeitsweise mit LPROEM 2-38
2.8.10.2. Arbeitsweise ohne LPROEM 2-39
2.8.11. Funktionsbeschreibung 2-39

2.9. BESCHREIBUNG DES PRÜFPROGRAMMES ZSPE20 2-39
2.9.1. Benennung 2-39
2.9.2. Dokumentationsnummer 2-39
2.9.3. Programmumfang 2-39
2.9.4. Speicherbedarf 2-40
2.9.5. Ausführungszeit 2-40

2.9.6. Testhilfsmittel/Geräteeinstellungen 2-40
2.9.7. Arbeit unter LPROEM 2-40
2.9.8. Arbeit ohne LPROEM 2-40
2.9.9. Generierinformationen 2-40
2.9.10. Fehlerinterpretation 2-41
2.9.10.1. Arbeitsweise mit LPROEM 2-41
2.9.10.2. Arbeitsweise ohne LPROEM 2-41
2.9.11. Funktionsbeschreibung 2-42

2.10. BESCHREIBUNG DES PRÜFPROGRAMMS ZTRAPI 2-42
2.10.1. Benennung 2-42
2.10.2. Dokumentationsnummer 2-42
2.10.3. Programmumfang 2-42
2.10.4. Speicherbedarf 2-43
2.10.5. Ausführungszeit 2-43
2.10.6. Testhilfsmittel/Geräteeinstellungen 2-43
2.10.7. Arbeit unter LPROEM 2-43
2.10.8. Arbeit ohne LPROEM 2-43
2.10.9. Generierinformationen 2-43
2.10.10. Fehlerinterpretation 2-44
2.10.10.1. Arbeitsweise mit LPROEM 2-44
2.10.10.2. Arbeitsweise ohne LPROEM 2-45
2.10.11. Funktionsbeschreibung 2-46

2.11. BESCHREIBUNG DES PRÜFPROGRAMMES ZBUSCI 2-47
2.11.1. Benennung 2-47
2.11.2. Dokumentationsnummer 2-47
2.11.3. Programmumfang 2-47
2.11.4. Speicherbedarf 2-47
2.11.5. Ausführungszeit 2-47
2.11.6. Testhilfsmittel/Geräteeinstellung 2-48
2.11.7. Arbeit unter LPROEM 2-48
2.11.8. Arbeit ohne LPROEM 2-48
2.11.9. Generierinformationen 2-48
2.11.10. Fehlerinterpretation 2-49
2.11.10.1. Arbeitsweise mit LPROEM 2-49
2.11.10.2. Arbeitsweise ohne LPROEM 2-51
2.11.11. Funktionsbeschreibung 2-51

2.12. BESCHREIBUNG DES PRÜFPROGRAMMS AKSAIS 2-52
2.12.1. Benennung 2-52
2.12.2. Dokumentationsnummer 2-52
2.12.3. Programmumfang 2-52
2.12.4. Speicherbedarf 2-53
2.12.5. Ausführungszeit 2-53
2.12.6. Testhilfsmittel/Geräteeinstellungen 2-53
2.12.7. Arbeit unter Leitprogramm 2-53
2.12.8. Arbeit ohne Leitprogramm 2-53
2.12.9. Generierinformationen 2-54
2.12.10. Fehlerinterpretation 2-56
2.12.10.1. Arbeitsweise mit LPROEM 2-56
2.12.10.2. Arbeitsweise ohne LPROEM 2-58
2.12.11. Funktionsbeschreibung 2-58

2.13. BESCHREIBUNG DES PRÜFPROGRAMMES GFLOPM 2-60
2.13.1. Benennung 2-60
2.13.2. Dokumentationsnummer 2-60
2.13.3. Programmumfang 2-60
2.13.4. Speicherbedarf 2-60
2.13.5. Ausführungszeit 2-60
2.13.6. Testgegenstand 2-60
2.13.7. Arbeit unter Leitprogramm LPROEM 2-61
2.13.8. Arbeit ohne Leitprogramm LPROEM 2-61
2.13.9. Generierinformationen 2-61
2.13.10. Fehlerinterpretation 2-63
2.13.11. Funktionsbeschreibung 2-65

2.14. KONSOLPRÜFPROGRAMMDISKETTE DS0 2-68
2.14.1. Leitprogramm LPROEM 2-68
2.14.2. Prüfprogramme für den Konsolrechner 2-69
2.14.3. Wichtige Generierungen für den Konsolrechner 2-70
2.14.3.1. Dauertestgenerierung NR.= 1 2-70
2.14.3.2. Dauertestgenerierung NR.= 2 2-71
2.14.3.3. Speichereinzeltestgenerierung 2-71
2.14.3.4. Dauertestgenerierung NR.= 3 2-72
2.14.3.5. Dauertestgenerierung NR.= 4 2-72
2.14.3.6. Formatiergenerierungen 2-73
2.14.3.7. Dauertestgenerierung NR.= 5 2-73
2.14.3.8. Dauertestgenerierung NR.= 6 2-74
2.14.3.9. Generierung für das Doppeln von Disketten 2-74
2.14.3.10. Generierung für das Vergleichen von Disketten 2-75

3. KONSOLSTEUERUNG UND BOOTSTRAPPING 3-1

3.1. ZUSAMMENFASSUNG 3-1

3.2. VORAUSSETZUNGEN 3-1

3.3. BEDIENUNGSHINWEISE 3-1

3.4. FUNKTIONSBESCHREIBUNG UND AUFBAU 3-4

3.5. KOMMANDOBESCHREIBUNG 3-5
3.5.1. Allgemeines 3-5
3.5.2. BOOT (B) 3-6
3.5.3. CLEAR (CL) 3-7
3.5.4. CONTINUE (C) 3-7
3.5.5. DEPOSIT (D) 3-8
3.5.6. DIRECTORY (DIR) 3-8
3.5.7. DISABLE DUAL (DIS DU) 3-9
3.5.8. DUPLICATE (DUP) 3-9
3.5.9. ENABLE DUAL (EN DUAL) 3-9
3.5.10. EXAMINE (E) 3-10

3.5.11. FORMAT (FO) 3-11
3.5.12. HALT (H) 3-11
3.5.13. HELP (HE) 3-11
3.5.14. INITIALIZE (I) 3-12
3.5.15. LINK (LI) 3-12
3.5.16. LOAD (L) 3-12
3.5.17. NEXT (N) 3-12
3.5.18. OVERLAY (O) 3-13
3.5.19. PERFORM (P) 3-13
3.5.20. QCLEAR (Q) 3-13
3.5.21. REBOOT (REB) 3-13
3.5.22. REPEAT (R) 3-13
3.5.23. SET (SE) 3-14
3.5.24. SHOW (SH) 3-15
3.5.25. START (S) 3-16
3.5.26. TEST (T) 3-16
3.5.27. TYPE (TY) 3-17
3.5.28. UNJAM (U) 3-17
3.5.29. WAIT DONE (WA D) 3-17
3.5.30. WCS (W) 3-17
3.5.31. Indirekt-Kommando (@) 3-17
3.5.32. CONTROL/P (^P) 3-18

3.6. BEDINGUNGEN FÜR DIE KOMMANDOVERARBEITUNG 3-18

3.7. KOMMENTARE INNERHALB VON KOMMANDOS 3-18

3.8. HELPFILE ZUM KONSOLPROGRAMM 3-19

3.9. HELPFILE FÜR ABKÜRZUNGEN DER KONSOLKOMMANDOSPRACHE 3-24

3.10. BOOTSTRAP-HELPFILE FÜR VMB.EXE 3-26

3.11. HELPFILE ERROR.HLP 3-30

4. M I K R O D I A G N O S E 4-1

4.1. ZUSAMMENFASSUNG 4-1

4.2. TESTBEDINGUNGEN 4-1
4.2.1. Hardware 4-1
4.2.2. Software 4-1

4.3. VORAUSSETZUNGEN 4-2

4.4. BEDIENUNGSHINWEISE 4-2

4.5. FUNKTIONSBESCHREIBUNG 4-2
4.5.1. Übersicht 4-2
4.5.2. Programmgröße (Speicherplatzbedarf und Speicheraufteilung) 4-2
4.5.3. Programmlaufzeit 4-4

4.5.4. Abarbeitungsbedingungen 4-4
4.5.5. Flag-Verwendung 4-4
4.5.6. Allgemeine Programmanforderungen 4-4
4.5.7. Fehlerermittlung 4-4
4.5.8. Programmitteilungen 4-4
4.5.9. Leistungsparameter des Testobjektes 4-4
4.5.10. Aktionen bei unerwarteten Fehlern 4-4
4.5.11. Testfolge bei mehreren Einheiten 4-4
4.5.12. Einstellung für Ablauf unter AFT 4-5
4.5.13. Testsektionen 4-5

4.6. TESTBESCHREIBUNG 4-5
4.6.1. Hardcoretest 4-5
4.6.1.1. Allgemeine Beschreibung 4-5
4.6.1.2. Struktur der Hardcoretests 4-6
4.6.2. Mikrotest 4-10
4.6.2.1. Allgemeine Beschreibung 4-10
4.6.2.2. Mikroteststruktur 4-11
4.6.3. Steuerung des Testablaufes 4-12
4.6.3.1. Allgemeine Beschreibung 4-12
4.6.3.2. Kommandos und Schalter 4-12
4.6.3.3. Fehlerausschriften 4-15
4.6.4. Interpretation der VBUS-Fehlerauschriften 4-17
4.6.4.1. Hardcore-Tests 4-17
4.6.4.2. Mikrotests 4-17
4.6.5.1. Fehlerfreier Testdurchlauf 4-19
4.6.5.2. Hardcoretestschleife 4-19
4.6.5.3. Mikrotest, Statische Messungen 4-21
4.6.5.4. Mikrotest, Dynamische Messungen 4-23

4.7. ABKÜRZUNGSVERZEICHNIS 4-23

4.8. AUFLISTUNG DER TESTS 4-24
4.8.1. Diskette DS2 4-24
4.8.1.1. Hardcoretests 4-24
4.8.1.2. Mikrotests 1 4-26
4.8.2. Diskette DS3 , Mikrotest 2 4-32

4.9. ANHANG
ZUORDNUNG TEST - FUNKTIONSGRUPPEN 4-39

5. P V K A A G R U N D B E F E H L S T E S T 5-1

5.1. ZUSAMMENFASSUNG 5-1

5.2. TESTBEDINGUNGEN 5-1
5.2.1. Hardware 5-1
5.2.2. Software 5-1

5.3. VORAUSSETZUNGEN 5-1

5.4. BEDIENUNGSHINWEISE 5-2

5.5. FUNKTIONSBESCHREIBUNG 5-4
5.5.1. Übersicht 5-4
5.5.2. Programmgröße 5-4
5.5.3. Programmlaufzeit und Programmitteilungen 5-4
5.5.4. Abarbeitungsbedingungen 5-5
5.5.5. Flag-Verwendung 5-5
5.5.6. Allgemeine Programmanforderungen 5-5
5.5.7. Fehlermitteilungen 5-5
5.5.8. Arbeit mit Unterbrechungspunkten 5-6
5.5.8.1. Setzen und Löschen von Unterbrechungspunkten 5-6
5.5.8.2. Erreichen eines Unterbrechungspunktes 5-7
5.5.8.3. Ausgabe der Tabelle der Unterbrechungspunkte 5-8

5.6. TESTBESCHREIBUNGEN 5-9
5.6.1. Test 1 - Test grundlegender Befehle 5-9
5.6.2. Test 2 - Tests bedingter und sonstiger Verzweigungsbefehle 5-9
5.6.3. Test 3 - Tests von Befehlen mit variabler Bitfeldverarbeitung 5-9
5.6.4. Test 4 - Adressierungstest 5-10
5.6.5. Test 5 - Operandenspezifikationstest 5-10
5.6.6. Test 6 - Test der Befehle AOBLSS und ACBL 5-10
5.6.7. Test 7 - Test der Befehle SOBGEQ und SOBGTR 5-10
5.6.8. Test 8 - Test der Befehle CASEB, CASEW und CASEL 5-10
5.6.9. Test 9 - Test der Integerarithmetik- und Logik-Befehle 5-10
5.6.10. Test 10 - Prozeduraufruftests 5-10
5.6.11. Test 11 - Tests sonstiger Befehle 5-10
5.6.12. Test 12 - Exception- und Interrupttests 5-11
5.6.13. Test 13 - Taktintervalltest 5-11
5.6.14. Test 14 - Test der Warteschlangenbefehle 5-11
5.6.15. Test 15 - Konsol-Transmitter-Test 5-11

5.7. AUFLISTUNG DER TESTS 5-11

6 GRUNDSPEICHERTEST 6-1

6.1. ZUSAMMENFASSUNG 6-1

6.2. TESTBEDINGUNGEN 6-1
6.2.1. Hardware 6-1
6.2.2. Software 6-1

6.3. VORAUSSETZUNGEN 6-2

6.4. BEDIENUNGSHINWEISE 6-2

6.5. FUNKTIONSBESCHREIBUNG 6-4
6.5.1. Übersicht 6-4
6.5.2. Programmgröße 6-5
6.5.3. Programmlaufzeit 6-5
6.5.4. Abarbeitungsbedingungen 6-5
6.5.5. Fehlerermittlung 6-6
6.5.6. Aktionen bei unerwarteten Fehlern 6-8
6.5.7. Programmitteilungen 6-8
6.5.8. Programmaufbau 6-9
6.5.9. Bedienerkommunikation und Testablaufsteuerung 6-9

6.6. TESTBESCHREIBUNG 6-12
6.6.1. Test 1 - Basistest 6-12
6.6.2. Test 2 - Marchtest 6-12
6.6.3. Test 3 - Refreshtest 6-12
6.6.4. Test 4 - Netzausfalltest 6-13

6.7. AUFLISTUNG DER TESTS 6-14

6.8. ABKÜRZUNGEN 6-14

7 PVKAB ARCHITEKTURBEFEHLSTEST 7-1

7.1. ZUSAMMENFASSUNG 7-1

7.2. TESTBEDINGUNGEN 7-1
7.2.1. Hardware 7-1
7.2.2. Software 7-1

7.3. VORAUSSETZUNGEN 7-1

7.4. BEDIENUNGSHINWEISE 7-1
7.4.1. Laden und Starten 7-1
7.4.2. Optionen 7-2
7.4.3. Testweiterführung 7-2

7.5. FUNKTIONSBESCHREIBUNG 7-2
7.5.1. Übersicht 7-2
7.5.1.1. Befehlstest-Service-Routine (RITS) 7-2
7.5.2. Programmgröße 7-2
7.5.3. Programmlaufzeit 7-3
7.5.4. Abarbeitungsbedingungen 7-3
7.5.5. Flagverwendung 7-3
7.5.6. Allgemeine Programmanforderungen 7-3
7.5.7. Fehlerermittlung 7-3
7.5.7.1. Grundfehlerausschriften 7-4
7.5.7.2. Erweiterte Fehlerausschriften 7-5
7.5.8. Programmitteilungen 7-5
7.5.9. Leistungsparameter des Testobjektes 7-5
7.5.10. Aktionen bei unerwarteten Fehlern 7-5

7.5.11. Testfolge bei mehreren Einheiten 7-5
7.5.12. Einstellungen für Ablauf unter AFT 7-6
7.5.13. Testsektionen 7-6
7.5.14. Verschiedenes 7-8
7.5.14.1. Übergehen eines Testfalles 7-8

7.6. TESTBESCHREIBUNG 7-9
7.6.1. Modulbeschreibung 7-9

7.7. ABKÜRZUNGSVERZEICHNIS 7-10

7.8. AUFLISTUNG DER TESTS 7-11
7.8.1. Die Tests des PVKABA 7-11
7.8.2. Die Tests des PVKABB 7-11
7.8.3. Die Tests des PVKABC 7-12
7.8.4. Test des PVKAD 7-12
7.8.5. Die Tests des PVKABE 7-12
7.8.6. Die Tests desPVKABF 7-12
7.8.7. Die Tests des PVKABG 7-12
7.8.8. Test des PVKABH 7-12

8. PVKAC GLEITKOMMABEFEHLSTEST 8-1

8.1. ZUSAMMENFASSUNG 8-1

8.2. TESTBEDINGUNGEN 8-1
8.2.1. Hardware 8-1
8.2.2. Software 8-1

8.3. VORAUSSETZUNGEN 8-1

8.4. BEDIENUNGSHINWEISE 8-1
8.4.1. Laden und Starten 8-1
8.4.2. Optionen 8-2

8.5. FUNKTIONSBESCHREIBUNG 8-2
8.5.1. Befehlstest-Service-Routine (RITS) 8-2
8.5.1.1. RIT Pass 0 Initialisierung 8-2
8.5.1.2. RIT Initialisierungsroutine 8-2
8.5.1.3. RIT Lösch-Routine 8-2
8.5.1.4. RITS 8-3
8.5.1.5. Testweiterführung 8-4
8.5.1.6. Accelerator-Test 8-4
8.5.2. Programmgröße 8-4
8.5.3. Programmlaufzeit 8-4
8.5.4. Abarbeitungsbedingungen 8-4
8.5.5. Flag-Verwendung 8-4
8.5.6. Allgemeine Programmanforderungen 8-5
8.5.7. Fehlermitteilungen 8-5
8.5.7.1. Einfache Fehlerausschrift 8-5
8.5.7.2. Erweiterte Fehlerausschrift 8-6
8.5.8. Programmitteilungen 8-7

8.5.9. Leistungsparameter des Testobjektes 8-7
8.5.10. Aktionen bei unerwarteten Fehlern 8-7
8.5.11. Testfolge bei mehreren Einheiten 8-7
8.5.12. Einstellungen für Ablauf unter AFT 8-7
8.5.13. Testsektionen 8-7
8.5.14. Verschiedenes 8-9
8.5.14.1. Übergehen eines Testfalles 8-9
8.5.14.2. First Part Done (FPD) 8-10

8.6. TESTBESCHREIBUNGEN 8-12

8.7. ABKÜRZUNGSVERZEICHNIS 8-13

8.8. AUFLISTUNG DER TESTS 8-14

9 PVKAD KOMPATIBILITÄTSMODUS-BEFEHLSTEST 9-1

9.1. ZUSAMMENFASSUNG 9-1

9.2. TESTBEDINGUNGEN 9-1
9.2.1. Hardware 9-1
9.2.2. Software 9-1

9.3. VORAUSSETZUNGEN 9-1

9.4. BEDIENUNGSHINWEISE 9-2

9.5. FUNKTIONSBESCHREIBUNG 9-2
9.5.1. Übersicht 9-2
9.5.2. Programmgröße 9-3
9.5.3. Programmlaufzeit 9-3
9.5.4. Abarbeitungsbedingungen 9-3
9.5.5. Flag-Verwendung 9-4
9.5.6. Allgemeine Programmanforderungen 9-4
9.5.7. Fehlerermittlung 9-4
9.5.8. Programmitteilungen 9-5
9.5.9. Leistungsparameter des Testobjekts 9-5
9.5.10. Aktionen bei unerwarteten Fehlern 9-5
9.5.11. Testfolge bei mehreren Einheiten 9-5
9.5.12. Einstellungen für Ablauf unter AFT 9-5
9.5.13. Testsektionen 9-5
9.5.13.1. Tests und Subtests 9-6
9.5.14. Verschiedenes 9-9

9.6. TESTBESCHREIBUNGEN 9-9
9.6.1. PVKAD0: Zentrale Module 9-9
9.6.1.1. Zentraler Programmdatenblock 9-9
9.6.1.2. Dispatch-Tabelle 9-9
9.6.1.3. Hardware-Parametertabelle 9-9
9.6.1.4. Globale Programmdaten 9-9
9.6.1.5. Programmausschriften 9-9
9.6.1.6. Initialisierung 9-10
9.6.1.7. Systemkontrollblock rücksetzen 9-10
9.6.1.8. Zusammenfassender Bericht 9-10
9.6.2. PVKAD1: Kompatibilitätsmodus-Exceptions 9-10

9.6.2. PVKAD1: Kompatibilitätsmodus-Exceptions 9-10
9.6.2.1. Test 1 Kompatibilitätsmodus, Eintritt und Austritt 9-10
9.6.2.2. Subtest 1.1. Exception durch TRAP-Befehle 9-11
9.6.2.3. Subtest 1.2. Exception durch T-Bit-TRAP 9-11
9.6.2.4. Subtest 1.3. Exception durch die Befehle RTT und RTI 9-11
9.6.2.5. Subtest 1.4. Exception durch ungerade Adresse 9-11
9.6.2.6. Subtest 1.5. Exception bei fehlerhafter Befehlskodierung 9-12
9.6.2.7. Subroutinen GEN/HNDLR 9-12
9.6.2.8. Subroutine ERR RPT 9-12
9.6.3. PVKAD2: SKR-Testroutinen für Test 1 9-12
9.6.4. PVKAD3: Teststeuerung Test 2, 3 und 4 9-13
9.6.4.1. Test 2 Steuerung des Tests der Einoperandenbefehle 9-13
9.6.4.2. Test 3-4 Steuerung des Tests der Zweioperandenbefehle bzw. Steuerbefehle 9-13
9.6.5. PVKAD4:Bedienungsroutinen für den Komatibilitätsmodus-Befehlstest 9-13
9.6.5.1. Allgemeine Bedienungsroutine für Kompatibilitätsmodus-Exceptions 9-13
9.6.5.2. Bedienungsroutine "RESET" 9-13
9.6.5.3. Bedienungsroutine "BPT" 9-14
9.6.5.4. Bedienungsroutine "IOT" 9-14
9.6.5.5. Bedienungsroutine "EMT" 9-14
9.6.5.6. Bedienungsroutine "TRAP" 9-14
9.6.5.7. TRAP bei fehlerhafter Adresse oder Codierung 9-14
9.6.5.8. Datenvorgabe für JMP und JSR 9-14
9.6.5.9. Datenvorgabe für Einoperandenbefehle 9-14
9.6.5.10. Datenvorgabe für Zweioperandenbefehle 9-15
9.6.5.11. Datenvorgabe Doppelwortbefehle 9-15
9.6.5.12. Datenvorgabe für STACK-MOVE 9-16
9.6.5.13. Dateneintrag in Speicher und Register 9-16
9.6.5.14. Datenprüfung 9-16
9.6.5.15. Laden eines Registers 9-17
9.6.5.16. Initialisieren aller Register 9-17
9.6.5.17. Fehlerausgabe 9-17
9.6.6. PVKAD5 SKR-Einoperandenbefehle 9-17
9.6.6.1. Test 2 (Subtest 1...Subtest 360) 9-17
9.6.6.2. Darstellung des Testprinzips 9-17
9.6.7. PVKAD6 SKR-Zweioperandenbefehle 9-19
9.6.7.1. Test 3 (Subtest 1...Subtest 206) 9-19
9.6.7.2. Darstellung des Testprinzips 9-19
9.6.8. PVKAD7 Steuerbefehle 9-20
9.6.8.1. Test 4 (Subtest 1...Subtest 46) 9-20
9.6.8.2. Subtest 1 Darstellung des Testprinzips für den BRANCH-Befehl 9-20
9.6.8.3. Subtest 2 Darstellung des Testprinzips am Beispiel BMI, BPL 9-21
9.6.8.4. Subtest 3...Subtest 20 Darstellung des Testprinzips am Beispiel BNE 9-22
9.6.8.5. Subtest 21...Subtest 43 Darstellung des Testprinzips am Beispiel JMP 9-22

9.6.8.6. Subtest 44 Darstellung des Testprinzips für den Befehl SOB 9-23
9.6.8.7. Subtest 45 Darstellung des Testprinzips für die Bedingungsflag-Behandlungs-Befehle 9-23
9.6.8.8. Subtest 46 Darstellung des Testprinzips für den Befehl NOP 9-24

9.7. ABKÜRZUNGSVERZEICHNIS 9-24

9.8. AUFLISTUNG DER TESTS 9-24

10. PVKAE TEST DER PRIVILEGIERTEN ARCHITEKTURBEFEHLE 10-1

10.1. ZUSAMMENFASSUNG 10-1

10.2. TESTBEDINGUNGEN 10-1
10.2.1. Hardware 10-1
10.2.2. Software 10-1

10.3. VORAUSSETZUNGEN 10-1

10.4. BEDIENUNGSHINWEISE 10-2
10.4.1. Laden und Starten 10-2

10.5. FUNKTIONSBESCHREIBUNG 10-2
10.5.1. Übersicht 10-2
10.5.2. Programmgröße 10-2
10.5.3. Programmlaufzeit 10-2
10.5.4. Abarbeitungsbedingungen 10-2
10.5.5. Flag-Verwendung 10-2
10.5.6. Allgemeine Programmanforderungen 10-2
10.5.7. Fehlerermittlung 10-3
10.5.8. Programmitteilungen 10-3
10.5.9. Leistungsparameter des Testobjektes 10-3
10.5.10. Aktionen bei unerwarteten Fehlern 10-3
10.5.11. Testfolge bei mehreren Einheiten 10-3
10.5.12. Einstellung für Ablauf unter AFT 10-3
10.5.13.1. Tests und Subtests 10-4
10.5.13.2. Testsektionen 10-4

10.6. TESTBESCHREIBUNGEN 10-4
10.6.1. Test 1 , Softwareinterrupts 10-4
10.6.1.1. Subtest 1.1 Software-Interrupt-Summary-Register (SISR) 10-4
10.6.1.2. Subtest 1.2 Software-Interrupt-Request-Register (SIRR) 10-5
10.6.2. Test 2 - Exceptions 10-6
10.6.2.1. Subtest 2.1 Exceptions bei reservierten Befehlen 10-6
10.6.2.2. Subtest 2.2 Exceptions bei privilegierten Befehlen 10-6
10.6.2.3. Subtest 2.3 Exceptions bei reservierten Adressierungsmodi

			10-6
10.6.3.	**Test 3 - REI-Befehlstest (return from exception or interrupt)**		**10-7**
	10.6.3.1.	Subtest 3.1 Keine Exceptions oder Interrupts	10-7
	10.6.3.2.	Subtest 3.2 Exceptions bei reservierten Operanden	10-7
	10.6.3.3.	Subtest 3.3 Trace-Traps	10-7
	10.6.3.4.	Subtest 3.4 Asynchrone Systemtraps (AST)	10-7
10.6.4.	**Test 4 - Test der Change-Mode-Befehle CHMx**		**10-8**
10.6.5.	**Test 5 - Prozessorregisterbefehle**		**10-8**
	10.6.5.1.	Subtest 5.1 Test des PME-Registers	10-8
	10.6.5.2.	Subtest 5.2 Test des SID-Registers (system identific.)	10-8
	10.6.5.3.	Subtest 5.3 Fault "reservierter Operand"	10-9
10.6.6.	**Test 6 - Test des Intervalltimers**		**10-9**
	10.6.6.1.	Subtest 6.1 Interrupt-enable-Bit	10-9
	10.6.6.2.	Subtest 6.2 Test des XFR- (XEGR-) Bits	10-9
	10.6.6.3.	Subtest 6.3 Einzeltakt für ICR	10-9
	10.6.6.4.	Subtest 6.4 ICR-Test	10-10
	10.6.6.5.	Subtest 6.5 Intervalltaktfolge	10-10
	10.6.6.6.	Subtest 6.6 Intervall-Clock-Überlauf	10-10
	10.6.6.7.	Subtest 6.7 Intervall-Clock ohne Interrupt	10-11
	10.6.6.8.	Subtest 6.8 Intervall-Clock mit Interrupt	10-11
	10.6.6.9.	Subtest 6.9 Weiterzählen des TODR (time of day register)	10-11
	10.6.6.10.	Subtest 6.10 "back to back loading" des TODR	10-12
	10.6.6.11.	Subtest 6.11 Schreiben/Lesen TODR	10-12
	10.6.6.12.	Subtest 6.12 Zählfunktion des TODR	10-12
10.6.7.	**Test 7 -Speicherverwaltung MM (memory management)**		**10-12**
10.6.8.	**Test 8 - Fehlerhafter Kernel-Stack**		**10-13**
	10.6.8.1.	Subtest 8.1 Zugriffsschutz bei ungültigem Kernel-Stack	10-13
	10.6.8.2.	Subtest 8.2 Befehl BPT zum KSP (kernel stack pointer)	10-14

10.6.9. Test 9 - Modusänderung zu schreibgeschütztem Stack 10-14
10.6.10. Test 10 - Test der Befehle PROBEx 10-15
10.6.10.1. Subtest 10.1 Test des Befehls PROBER 10-15
10.6.10.2. Subtest 10.2 Test des Befehls PROBEW 10-15
10.6.11. Test 11 - Test der Befehle SVPCTX und LDPCTX 10-15
10.6.11.1. Subtest 11.1 SVPCTX mit normalen Daten, Interruptstack 10-15
10.6.11.2. Subtest 11.2 SVPCTX mit normalen Daten, Kernelstack 10-16
10.6.11.3. Subtest 11.3 SVPCTX mit extremen Daten ("alles 0") 10-16
10.6.11.4. Subtest 11.4 SVPCTX mit extremen Daten ("alles 1") 10-16
10.6.11.5. Subtest 11.5 LDPCTX mit normalen Daten, Kernel-Stack 10-16
10.6.11.6. Subtest 11.6 LDPCTX mit normalen Daten vom Interrupt-Stack zum Kernel-Stack 10-16
10.6.11.7. Subtest 11.7 LDPCTX mit reservierten Operanden 10-16
10.6.11.8. Subtest 11.8 LDPCTX normal im AST-Level 10-16
10.6.11.9. Subtest 11.9 wie Subtest 11.7 mit "0"-Feld 3 10-17

10.7. ABKÜRZUNGSVERZEICHNIS 10-17

10.8. AUFLISTUNG DER TESTS 10-18

11. PVRAM: SPEICHERTEST-PROGRAMM 11-1

11.1. ZUSAMMENFASSUNG 11-1

11.2. TESTBEDINGUNGEN 11-1
11.2.1. Hardware 11-1
11.2.2. Software 11-1

11.3. VORAUSSETZUNGEN 11-1

11.4. BEDIENUNGSHINWEISE 11-2

11.5. FUNKTIONSBESCHREIBUNG 11-2
11.5.1. Übersicht 11-2
11.5.2. Programmgröße 11-2
11.5.3. Programmlaufzeit 11-2
11.5.4. Abarbeitungsbedingungen 11-2
11.5.5. Flag-Verwendung 11-3
11.5.6. Allgemeine Programmanforderungen 11-3
11.5.7. Fehlermitteilungen 11-3
11.5.8. Programmitteilungen 11-3
11.5.9. Leistungsparameter des Testobjekts 11-3
11.5.10. Aktionen bei unerwarteten Fehlern 11-4
11.5.11. Testfolge bei mehreren Einheiten 11-4
11.5.12. Einstellungen für Ablauf unter AFT 11-4
11.5.13. Testsektionen 11-4
11.5.14. Verschiedenes 11-4

11.6. TESTBESCHREIBUNGEN 11-4
11.6.1. Test 1 (MEMFT) 11-4
11.6.2. Test 2 (CONFT) 11-4

11.7. ABKÜRZUNGSVERZEICHNIS 11-5

11.8. AUFLISTUNG DER TESTS 11-5

12. PVSBA - K 1840 AUTOSIZER-TEST 12-1

12.1. ZUSAMMENFASSUNG 12-1

12.2. TESTBEDINGUNGEN 12-1
12.2.1. Hardware 12-1
12.2.2. Software 12-1

12.3. VORAUSSETZUNGEN 12-1

12.4. BEDIENUNGSHINWEISE 12-2

12.5. FUNKTIONSBESCHREIBUNG 12-2
12.5.1. Übersicht 12-2
12.5.2. Programmgröße 12-2
12.5.3. Programmlaufzeit 12-2
12.5.4. Abarbeitungsbedingungen 12-2
12.5.5. Flagverwendung 12-3
12.5.6. Allgemeine Programmanforderungen 12-3
12.5.7. Fehlermitteilung 12-3
12.5.8. Programmitteilungen 12-3
12.5.9. Leistungsparameter des Testobjektes 12-3
12.5.10. Aktionen bei unerwarteten Fehlern 12-3
12.5.11. Testfolge bei mehreren Einheiten 12-4
12.5.12. Einstellungen für Ablauf unter AFT 12-4
12.5.13. Testsektionen 12-4

12.6. TESTBESCHREIBUNGEN 12-4
12.6.1. Test 01: DEFAULT MODE
Standard-Modus 12-4
12.6.2. Test 02: SELFTEST MODE
Eigentest-Modus 12-5
12.6.2.1. ATTACH 12-5
12.6.2.2. CHANGE 12-5
12.6.2.3. EXIT 12-9
12.6.2.4. HELP 12-10
12.6.2.5. LIST 12-12
12.6.2.6. SIZE 12-12
12.6.2.7. READ 12-12
12.6.2.8. WRITE 12-14
12.6.3. Test 03: MANUAL MODE
Manual-Modus 12-15

12.7. AUFLISTUNG DER TESTS 12-17

13. PVCAA MSBUS-ADAPTERTEST 13-1

13.1. ZUSAMMENFASSUNG 13-1

13.2. TESTBEDINGUNGEN 13-1
13.2.1. Hardware 13-1
13.2.2. Software 13-1

13.3. VORAUSSETZUNGEN 13-1

13.4. BEDIENUNGSHINWEISE 13-1

13.5. FUNKTIONSBESCHREIBUNG 13-2
13.5.1. Übersicht 13-2
13.5.2. Programmgröße 13-2
13.5.3. Programmlaufzeit 13-3
13.5.4. Abarbeitungsbedingungen 13-3
13.5.5. Flagverwendung 13-3
13.5.6. Allgemeine Programmanforderungen 13-3
13.5.7. Fehlerermittlung 13-3
13.5.7.1. Register-Hard-Fehler 13-4
13.5.7.2. Map-Hard-Fehler 13-4
13.5.7.3. Datenvergleichs-Hard-Fehler 13-4
13.5.8. Programmitteilungen 13-5
13.5.9. Leistungsparamter des Testobjektes 13-5
13.5.10. Aktionen bei unerwarteten Fehlern 13-5
13.5.11. Testfolge bei mehreren Einheiten 13-5
13.5.12. Einstellungen für Ablauf unter AFT 13-6
13.5.13. Testsektionen 13-6

13.6. TESTBESCHREIBUNG 13-6
13.6.1. Test 1 - Interner Busbestätigungstest 13-6
13.6.2. Test 2 - Initialisierungstest 13-6
13.6.3. Test 3 - Steuerregistertest 13-7
13.6.3.1. Subtest 1 13-7
13.6.3.2. Subtest 2 13-8

13.6.4.	**Test 4 - Diagnose-Registertest**	**13-8**
	13.6.4.1. Subtest 1	13-8
	13.6.4.2. Subtest 2	13-8
	13.6.4.3. Subtest 3	13-8
	13.6.4.4. Subtest 4	13-8
	13.6.4.5. Subtest 5	13-9
13.6.5.	**Test 5 - Test des Registers für virtuelle Adresse**	**13-9**
13.6.6.	**Test 6 - Map-Registertest**	**13-9**
	13.6.6.1. Subtest 1	13-9
	13.6.6.2. Subtest 2	13-9
	13.6.6.3. Subtest 3	13-10
13.6.7.	**Test 7 - Test der Datenwege externer Register**	**13-10**
13.6.8.	**Test 8 - MBDIB-Diagnose-Registertest**	**13-10**
13.6.9.	**Test 9 - Statusregistertest**	**13-11**
	13.6.9.1. Subtest 1	13-12
	13.6.9.2. Subtest 2	13-12
	13.6.9.3. Subtest 3	13-12
	13.6.9.4. Subtest 4 und Subtest 5	13-12
	13.6.9.5. Subtest 6 bis 9	13-13
	13.6.9.6. Subtest 10	13-13
	13.6.9.7. Subtest 11	13-13
13.6.10.	**Test 10 - Test der Mapping-Funktion**	**13-14**
	13.6.10.1. Subtest 1	13-14
	13.6.10.2. Subtest 2	13-14
13.6.11.	**Test 11 - Übertragungstest für das Statusregister**	**13-15**
	13.6.11.1. Subtest 1	13-15
	13.6.11.2. Subtest 2 bis Subtest 9	13-15
13.6.12.	**Test 12 - Blockübertragungstest**	**13-17**
	13.6.12.1. Subtest 1	13-17
	13.6.12.2. Subtest 2	13-17
13.6.13.	**Test 13 - Bytezählertest**	**13-17**
	13.6.13.1. Subtest 1	13-17
	13.6.13.2. Subtest 2	13-17
	13.6.13.3. Subtest 3	13-17
	13.6.13.4. Subtest 4	13-18
13.6.14.	**Test 14 - READ/WRITE-Funktionstest**	**13-18**
	13.6.14.1. Subtest 1	13-18
	13.6.14.2. Subtest 2 bis Subtest 5	13-18
13.6.15.	**Test 15 - SILO-Test**	**13-18**
	13.6.15.1. Subtest 1	13-19
	13.6.15.2. Subtest 2	13-19
13.6.16.	**Test 16 - Interrupttest**	**13-19**
	13.6.16.1. Subtest 1	13-19
	13.6.16.2. Subtest 2	13-20
	13.6.16.3. Subtest 3	13-20
	13.6.16.4. Subtest 4	13-20
	13.6.16.5. Subtest 5	13-20
	13.6.16.6. Subtest 6	13-21
	13.6.16.7. Subtest 7	13-21
	13.6.16.8. Subtests 8, 9 und 11	13-21
	13.6.16.9. Subtest 10	13-21
	13.6.16.10. Subtest 12	13-22
	13.6.16.11. Subtest 13 und Subtest 14	13-22
	13.6.16.12. Subtest 15	13-22

13.6.17. Test 17 - Erweiterter MCP-Test 13-22
13.6.17.1. Subtest 1 13-22
13.6.17.2. Subtest 2 13-22
13.6.18. Test 18 13-23
13.6.19. Test 19 - RH11-TB Tests 13-23
13.6.19.1. Subtest 1 13-23
13.6.19.2. Subtest 2 13-23
13.6.19.3. Subtest 3 13-24
13.6.19.4. Subtest 4 13-24
13.6.19.5. Subtest 5 13-24
13.6.19.6. Subtest 6 13-24
13.6.19.7. Subtest 7 13-24
13.6.19.8. Subtest 8 bis Subtest 15 13-25
13.6.20. Test 20 - Test des Ladens des VAR 13-25
13.6.21. Registerbeschreibung 13-25
13.6.21.1. Interne Prozessorregister 13-25
13.6.21.2. MBA interne Register 13-26
13.6.21.3. MBE-Register 13-28

13.7. ABKÜRZUNGSVERZEICHNIS 13-29

13.8. AUFLISTUNG DER TESTS 13-29

14. TEST DES SKRBUS-ADAPTERS 14-1

14.1. ZUSAMMENFASSUNG 14-1

14.2. TESTBEDINGUNGEN 14-1
14.2.1. Hardware 14-1
14.2.2. Software 14-1

14.3. VORAUSSETZUNGEN 14-1

14.4. BEDIENUNGSHINWEISE 14-2

14.5. FUNKTIONSBESCHREIBUNG 14-2
14.5.1. Übersicht 14-2
14.5.2. Programmgröße 14-2
14.5.3. Programmlaufzeit 14-3
14.5.4. Abarbeitungsbedingungen 14-3
14.5.5. Flagverwendung 14-3
14.5.6. Allgemeine Programmanforderungen 14-3
14.5.7. Fehlerermittlung 14-3
14.5.8. Programmitteilungen 14-4
14.5.9. Leistungsparameter des Testobjekts 14-4
14.5.10. Aktionen bei unerwarteten Fehlern 14-4
14.5.11. Testfolgen bei mehreren Einheiten 14-5
14.5.12. Einstellungen für Ablauf unter AFT 14-5
14.5.13. Testsektionen 14-5
14.5.14. Verschiedenes 14-5

14.6. TESTBESCHREIBUNGEN 14-5
14.6.1. Test 1 - CNFGR-Registerprüfung 14-5
14.6.2. Test 2 - DCR-Register-Test 14-5
14.6.3. Test 3 - Test DCR und CNFGR 14-6
14.6.4. Test 4 - Test UACR-Register 14-6
14.6.5. Test 5 - Test UASR-Register 14-6
14.6.6. Test 6 - Initialisierungstest 14-7
14.6.7. Test 7 - SKRBUS-Netzausfall-Test 14-7
14.6.8. Test 8 - ADINIT Taktierung und Netzausfall. 14-8
14.6.9. Test 9 - Interrupt-Logik 14-8
14.6.10. Test 10 - Test SKRBUS-Mask-Error-Funktion. 14-8
14.6.11. Test 11 - Test SBA-Mask-Error-Funktion. 14-9
14.6.12. Test 12 - Map-Register-Test 14-9
14.6.13. Test 13 - Map-Register-Test 14-9
14.6.14. Test 14 - BRSVR-Register-Test 14-10
14.6.15. Test 15 - SSYNTO-Bit-Test 14-10
14.6.16. Test 16 - FUBAR-Register-Test 14-11
14.6.17. Test 17 - FMER-Register-Test 14-11
14.6.18. Test 18 - DPR-Register-Test 14-12
14.6.19. Test 19 - Map-Register-Disable-Funktion 14-12
14.6.20. Test 20 - Status-Bit-Test 14-13
14.6.21. Test 21 - Datentransfer-Test /Schreiben. 14-13
14.6.22. Test 22 - Test Purge doing a NOP 14-14
14.6.23. Test 23 - DPR-Mask-Bit-Test 14-14
14.6.24. Test 24 - Adressleitungstest 14-15
14.6.25. Test 25 - BDP-Adresstest 14-15
14.6.26. Test 26 - Test der BUSY-Bestätigungs-Funktion 14-15
14.6.27. Test 27 - SKRBUS-Schreiben 14-16
14.6.28. Test 28 - Mikro-Controller-Test 14-16
14.6.29. Test 29 - Mikro-Controller-Test 14-16
14.6.30. Test 30 - Datenübertragungstest - Lesen. 14-17
14.6.31. Test 31 - Mikro-Controller-Test 14-17
14.6.32. Test 32 - Map-Register-Test 14-18
14.6.33. Test 33 - Map-Register-Test 14-18
14.6.34. Test 34 - Ein-/Ausgabe-DATO-Funktion 14-19
14.6.35. Test 35 - E/A-DATO/B-Funktion 14-19
14.6.36. Test 36 - Test E/A-DATI (BO1) 14-19
14.6.37. Test 37 - Test E/A-DATI-(B23)-Funktion 14-19
14.6.38. Test 38 - Interlock-Logik und DAT-I/P-Test 14-20
14.6.39. Test 39 - Test Befehls-Adressen Fehler 14-20
14.6.40. Test 40 - Test Befehls-Adress-Time-out-Fehler 14-20
14.6.41. Test 41 - Lost-Error-Bit-Funktion 14-21
14.6.42. Test 42 - DEF.-, MAP.-, PAR.-Funktion 14-22
14.6.43. Test 43 - ISB-Interrupt-Test 14-22
14.6.44. Test 44 - ISB-Interrupt-Test 14-23
14.6.45. Test 45 - ISB-Interrupt-Test 14-24
14.6.46. Test 46 - FMER-Register-Test nach Fehlern 14-25
14.6.47. Test 47 - Test SET DATI-Funktion 14-26
14.6.48. Test 48 - Test SET DAT-I/P-Funktion 14-26
14.6.49. Test 49 - Test SET DATO-Funktion 14-26
14.6.50. Test 50 - Test SET DATO/B-Funktion 14-27
14.6.51. Test 51 - ISB-Interrupt durch den SET 14-27
14.6.52. Test 52 - Zugriff während der Ausschaltfolge 14-28

14.6.53. Test 53 - Disable-Interrupt-Bit und Ablaufblockierung 14-28
14.6.54. Test 54 - Contention-Logik-Test 14-29
14.6.55. Abkürzungsverzeichnis 14-29
14.6.56. Auflistung der Tests 14-30

15. BUS-INTERAKTION-TEST 15-1

16. K 1840 SYSTEM-TEST 16-1

17 DIAGNOSE-SUPERVISOR 17-1

17.1. EINFÜHRUNG 17-1

17.2. DIAGNOSE- UND SUPERVISOR-KOMMANDOS 17-2
17.2.1. Programm- und Testablaufsteuerkommandos 17-2
17.2.2. Scripting 17-14
17.2.3. Abarbeitungssteuerfunktionen 17-16
17.2.4. Debug- und Dienstprogramm-Kommandos 17-20

17.3. DIAGNOSEPROGRAMME UNTER SUPERVISOR 17-24
17.3.1. Starten des Supervisors im User-Modus 17-24
17.3.2. Starten des Supervisors im Stand-alone-Modus 17-25
17.3.2.1. Rechner einschalten 17-25
17.3.2.2. Booten des Supervisors von der Systemplatte 17-26
17.3.2.3. Booten des Supervisors von der Diskette 17-27
17.3.2.4. Nutzung von Script-Dateien 17-27

17.4. ÜBERSICHT ÜBER DIE SUPERVISORKOMMANDOS 17-30

ANLAGE A BEGRIFFSERKLÄRUNG A-1

ANLAGE B DISKETTENÜBERSICHT B-1

B I L D E R

1-1:	Übersichtsblockdiagramm	1-1
1-2:	Struktur der Diagnose-Umgebung	1-3
1-3:	Schematische Darstellung des RVS K 1840 - Diagnosesystems	1-4
1-4:	Laden von Diagnoseprogrammen und deren Steuerung	1-7

T A B E L L E N

2-1: Prüfprogramme für den Konsolrechner 2-2
2-2: Tauschreihenfolge der STE bei auftretendem Fehler 2-3

1. Übersicht

1.1. Diagnosesystemstruktur des RVS K 1840

1.1.1. Allgemeines

Das Rechnersystem mit virtuellem Speicher RVS K 1840 umfaßt einen zentralen Prozessor (CPU), ein Hauptspeichersubsystem, ein E/A-Subsystem, ein Konsolsubsystem und periphere Geräte.
Die Struktur des RVS K 1840 zeigt Bild 1-1

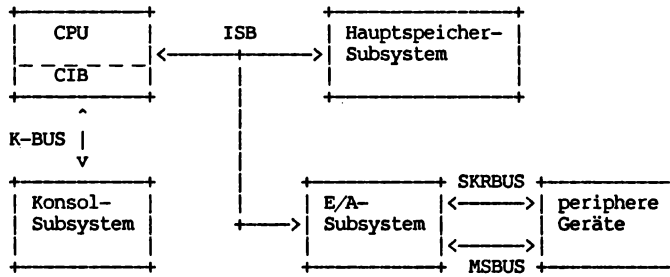

Bild 1-1: Übersichtsblockdiagramm

Der zentrale Prozessor (CPU) ist ein mikroprogrammierter 32-bit-Rechner hoher Geschwindigkeit. Es sind zwei mikroprogrammierte Befehlssätze integriert (K 1840-Befehlssatz und SKR-Befehlssatz).
Die CPU arbeitet über einen internen Synchronbus (ISB) mit dem Hauptspeicher-Subsystem und dem E/A-Subsystem zusammen.
Der ISB ist Teil des E/A-Subsystems, das auch den MSBUS-Adapter und den SKRBUS-Adapter enthält. Der MSBUS verbindet die Geräte hoher Datenübertragungsrate zum Rechnersystem. Der SKRBUS verbindet alle anderen peripheren Geräte zum Rechnersystem. Das E/A-Subsystem leistet Datenpufferung und Transferfunktionen.
Das Konsolsubsystem stellt die Verbindung des Operators über den K-BUS zur CPU dar. Vom Konsolsubsystem können z.B. das Betriebssystem
SVP 1800 und der Diagnose-Supervisor gebootet werden. Auch können die Diagnoseprogramme vom Konsolterminal bedient werden.
Dieser Hardwarestruktur entsprechend ist das Diagnosesystem des RVS K 1840 aufgebaut. Es besteht aus einer Hierarchie von Diagnoseprogrammen und ist dem Serviceingenieur und dem Nutzer ein leistungsfähiges Werkzeug bei der Prüfung der Hardware, bei der Fehlersuche und Wartung des gesamten Rechnersystems.
Fehlererkennung und -isolation durch das Diagnosesystem beschleunigen die Reparaturzeit beträchtlich.
Die Diagnosesystemhierarchie besteht aus sechs Diagnoseebenen, die sich unterscheiden in:

- ihrer Funktion,
  hohe Ebenennummer bedeutet hohe Testschärfe bezüglich einer bestimmten Baugruppe (hohe Hardwarenähe, Test spezifischer Funktionen),
  niedrige Ebenennummer bedeutet geringe Testschärfe (geringe Hardwarenähe, Test allgemeiner Funktionen)

- der Umgebung, in der die Diagnoseprogramme arbeiten
  Einige Programme z.B. arbeiten parallel zum Nutzerbetrieb (in Nutzerumgebung). Andere Programme erfordern die ausschließliche Nutzung des Rechnersystems, d.h. sie arbeiten stand-alone (in Konsolumgebung, CPU-Clusterumgebung usw.)

- den verwendeten Kommandosätzen.
  So gelten z.B. für die Konsolebene andere Kommandosätze als für die Ebenen 3, 2, 2R.

Darüber hinaus existieren für intelligente Peripheriegeräte interne Diagnoseprogramme für den Selbsttest (z.B. bei Netzeinschaltung). Diese Programme sind den entsprechenden Gerätedokumentationen zugeordnet und nicht Gegenstand dieses Handbuches.

1.1.2. Ebenen des Diagnosesystems

Ebene 1: Auf dem Betriebssystem SVP 1800 basierende Hilfsmittel zur Prüfung des Gesamtsystems von Hard- und Software (Nutzung virtueller QIO).
Dazu gehören die dem SVP zugeordneten Programmpakete
. UETP (User Environment Test Package)
. SDA (System Dump Analyzer)
Näheres siehe Kapitel 1.6.

Ebene 2R: Auf Supervisor aufbauende Diagnoseprogramme, die nur unter SVP arbeiten (Nutzung physischer QIO). Dies sind Diagnoseprogramme, die nicht vom Supervisor im stand-alone-Modus unterstützt werden und der Systemexerciser. Der Supervisor dient der Steuerung der Diagnoseprogramme der Ebenen 2R, 2 und 3 und liefert den Kontext für die Abarbeitung der Diagnoseprogramme dieser Ebenen.

Ebene 2: Auf Supervisor aufbauende Diagnoseprogramme, die sowohl Stand-alone als auch unter SVP 1800 arbeiten können (Nutzung physischer QIO).
Dazu gehören:
. Bus interaction-Programm
. Befehlstests
. Formatierungsprogramme für Plattenspeicher
. Programme für Zuverlässigkeitstests

Ebene 3: Auf Supervisor aufbauende Diagnoseprogramme, die nur Stand-alone arbeiten können.
Dazu gehören:
. Diagnoseprogramme für die funktionelle Prüfung von peripheren Geräten
. Diagnoseprogramme für Reparaturen an peripheren Geräten
. Befehlstests
. Busadaptertests usw.

Ebene 4: Stand-alone arbeitende Macrodiagnoseprogramme ohne Supervisorsteuerung. Dies ist z.B. der Grundbefehlstest. Er testet die

Grundfunktionen der CPU, die für die Abarbeitung des Supervisors erforderlich sind.

Ebene 5: Auf der Konsole aufbauende Diagnoseprogramme, die nur Stand-alone arbeiten können. Dazu gehören:
. Mikrodiagnoseprogramme
Konsolprogramm
ROM-residente Tests, die nach Netzzuschaltung abgearbeitet werden.
Diagnoseprogramme zum Test des Konsolrechners und seiner peripheren Geräte

1.1.3. Struktur der Diagnose-Umgebung

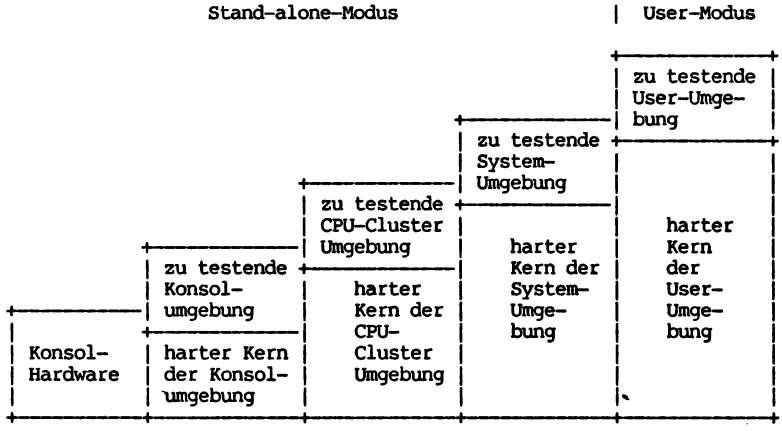

Bild 1-2: Struktur der Diagnose-Umgebung

Die o.g. sechs Programmebenen arbeiten gemäß Bild 1-2 im Kontext von vier Umgebungen:
- der Userumgebung
- der Systemumgebung
- der Clusterumgebung
- der Konsolumgebung

Jede Umgebung besteht aus einem Teil Hardware und der dazugehörigen Firmware. Diese vier Umgebungen wiederum arbeiten innerhalb von zwei Operationsmodi (Online-Modus, d.h. mit SVP, und Stand-alone-Modus, d.h. ohne SVP).
Bild 1-2 zeigt außerdem, daß eine Überlappung zwischen den zu testenden Feldern in den verschiedenen Diagnose-Umgebungen vorhanden ist.
Für jede Umgebung bildet ein Teil der Hardware den "Harten Kern", von dem vorausgesetzt wird, daß er fehlerfrei arbeitet. Die speziellen Diagnoseprogramme testen die Hardware, die diesen "Harten Kern" umgibt.
Bild 1-3 gibt eine schematische Darstellung dieser Beziehungen zwischen Operationsmodus, Diagnoseumgebung und -ebene.

<table>
<tr><td rowspan="2">online</td><td rowspan="2">User-
Umgebung</td><td rowspan="2">Diagnose-</td><td>Ebene 1</td></tr>
<tr><td>Ebene 2R</td></tr>
<tr><td rowspan="3">stand-alone

(nur Konsol-
terminal)</td><td>System-
Umgebung</td><td rowspan="2">Supervisor</td><td>Ebene 2</td></tr>
<tr><td rowspan="2">CPU
Cluster-Umgebung</td><td>Ebene 3</td></tr>
<tr><td>Ebene 4</td></tr>
<tr><td></td><td colspan="2">Konsol-Umgebung</td><td>Konsol-Ebene</td></tr>
</table>

Bild 1-3: Schematische Darstellung des RVS K 1840 - Diagnosesystems

- Konsolumgebung
  Die Konsolumgebung liefert die fundamentalen Operator-, Debugging- und Rechnerbasis-Diagnosefunktionen. Der "Harte Kern" der Konsolumgebung besteht aus Konsolhardware und dem Terminal.
  Die zu testende Hardware der Konsolumgebung besteht aus den Teilen der CPU-Hardware, des SKRBUS-Adapters bzw. MSBUS-Adapters und des Speichers, die von den Mikroprogrammtests erreicht werden können.
  Die Konsolumgebung arbeitet nur stand-alone. Der Operator bedient das System vom Konsolterminal.

- CPU-Cluster-Umgebung
  Ebenso wie die Konsolumgebung arbeitet die CPU-Cluster-Umgebung Stand-alone. Der Operator muß das Konsolterminal nutzen.
  Die CPU-Cluster-Umgebung besteht aus folgenden Rechnerkomponenten:
  komplette CPU
  Speicher
  . E/A-Kanäle
  Die Hardware, die in der Konsolumgebung von den Mikrodiagnosen getestet wird, bildet den "Harten Kern" der CPU-Cluster-Umgebung.
  Die CPU-Cluster-Umgebung liefern die Diagnoseprogramme der Ebene 4 und eine kleine Anzahl von Diagnoseprogrammen der Ebene 3. Diese Programme testen die Grund- und erweiterten CPU-, die Speicher- und die E/A-Kanalfunktionen. Die E/A-Kanalfunktionstests basieren auf E/A-Kurzschlußtests.

- Systemumgebung
  Auch die Systemumgebung arbeitet stand-alone. Sie beinhaltet ein großes Spektrum von Diagnoseprogrammen der Ebenen 3 und 2. Die Diagnosestrategie sieht für jedes E/A-Subsystem eine Reihe von Programmen unterschiedlicher Aufgabenstellung und Testschärfe vor. Die Aufgabenstellung erstreckt sich von Tests der Logik und E/A-Schleifentests (Ebene 3) über die Tests der Grundfunktionen und Zeitmessung (Ebene 3 oder 2) bis zu Zuverlässigkeitstests und Parallelarbeit mehrerer Geräte (Ebene 2). Die in der Cluster-Umgebung getestete Hardware bildet den "Harten Kern" für die Systemumgebung.

- User-Umgebung
  In der User-Umgebung arbeiten die Diagnoseprogramme im online-Modus unter SVP 1800. Der Operator kann die Diagnoseprogramme von jedem Terminal des Systems, einschließlich dem Konsolterminal bedienen.
  Die User-Umgebung schließt sowohl die Diagnoseprogramme der Ebene 2, als auch die in der Ebene 2R ein. Viele der Diagnoseprogramme, die in der User-Umgebung laufen, können simultan zu den Anwenderprogrammen arbeiten. Einige jedoch, wie das Systemdiagnoseprogramm, fordern die ausschließliche Nutzung des Rechnersystems.

- Reparaturniveau und Funktionsniveau
  Die Stellung des Programmes in der Hierarchie drückt sich sowohl in der diesem Programm zugeordneten Ebene (Ebene 1 bis 5) als auch in der Zuordnung der zu testenden Umgebung und in der Unterscheidung nach Programmen für Reparaturzwecke (Reparaturniveau) und Programmen für Funktionstests (Funktionsniveau) aus. Programme des Reparaturniveaus sind mehr hardwarebezogene Testprogramme als die der Funktionsniveaus.
  Die Unterscheidung nach Funktions- und Reparaturniveau erfolgt nur für supervisorgesteuerte Programme (Ebene 3, 2, 2R). Jedes Programm ist einem der beiden Niveaus (Reparatur- oder Funktionsniveau) zugeordnet. Die meisten Diagnoseprogramme sind vom Funktionsniveau. Für E/A-Subsysteme existieren Programme im Reparatur- und im Funktionsniveau. Die Zuordnung ist entweder aus dem Programmnamen oder aus dem HELP-File für das jeweilige Diagnoseprogramm zu erkennen.

  o Reparaturniveau:
    Allgemein kann man sagen, daß Programme des Reparaturniveaus größere logische Komplexe testen und bei Fehlererkennung den fehlerhaften Modul und die fehlerhafte Funktion ausschreiben.
    Peripherie-Diagnoseprogramme der Ebene 3 für Reparaturzwecke z.B. prüfen das Interface zwischen Kontroller und E/A-Bus, testen die Kontrollerkommandos und damit die mit jedem Kommando verbundene Logik. Haben die zu testenden Geräte Wartungsmodus, ist dieser im Programmtest integriert. Fehler werden vom Programm provoziert und die Reaktion des Gerätes geprüft.
    Weiterhin werden die Datenübertragungsfunktionen geprüft. Beim Test intelligenter Geräte wird, wenn erforderlich, ein spezieller Mikrodiagnosekode in den internen Speicher des Gerätes geladen, um die Logik des Gerätes zu prüfen.
    Der Standardgerätemikrokode des Gerätes bildet die Grundlage für die weiteren Tests des intelligenten Gerätes.

  o Funktionsniveau:
    Programme des Funktionsniveaus testen den größten Teil der Funktionen des Gerätes, und im Fehlerfall wird die fehlerhafte Funktion ausgeschrieben.
    Peripherie-Diagnoseprogramme der Ebene 3 für Funktionstests z.B. prüfen im wesentlichen die Funktionen, die zu den Testprogrammen der Ebene 2 transparent sind, durch Direkt-E/A. Indem sie Teile des Drivers simulieren, schaffen sie eine Brücke zu den Programmen der Ebene 2. Es werden alle Datenübertragungskommandos für alle Datentypen und alle elektromechanischen Funktionen auf Einhaltung ihrer spezifischen Parameter und Zeitbedingungen geprüft. Darüber hinaus werden alle Funktionen geprüft, die vom Operator im Usermodus auszuführen sind.
    Die Diagnoseprogramme der Ebene 2 sind meist Funktionsdiagnoseprogramme. Sie testen also auch alle Funktionen und Funktions-Kombinationen mit einer Vielzahl von Datenkombinationen. Weil Programme der Ebene 2 sowohl online als auch offline arbeiten können, wurden die Zuverlässigkeitstests für Geräte hier eingeordnet.

Wie man aus dem bisher gesagten erkennt, existieren im Diagnosesystem des RVS K 1840 mehrere aufeinander aufbauende Programmpakete, für deren Steuerung verschiedene Monitore erforderlich sind.

- Konsoltest	Ebene 5
- Mikrodiagnosen	Ebene 5
- Grundbefehlstest, Grundspeichertest	Ebene 4
- Diagnoseprogramme unter Supervisorsteuerung	Ebenen 3,2,2R
- im SVP 1800 integrierte Testhilfsmittel (UETP, SDA)	Ebene 1

1.1.4. Laden der Diagnoseprogramme und deren Steuerung (Übersicht)

Das Diagnosesystem für RVS K 1840 hat eine große Flexibilität bezüglich Laden und Steuerung der verschiedenen Programmebenen.
Z.B. können die Programme der Ebenen 2 und 3 sowohl vom Folienspeicher der Konsole als auch vom Plattenspeicher geladen werden. Die Mikrodiagnosen werden gewöhnlich nur vom Folienspeicher der Konsole geladen.
Bild 1-4 zeigt die Lademöglichkeiten der verschiedenen Diagnoseebenen und deren Steuerung.

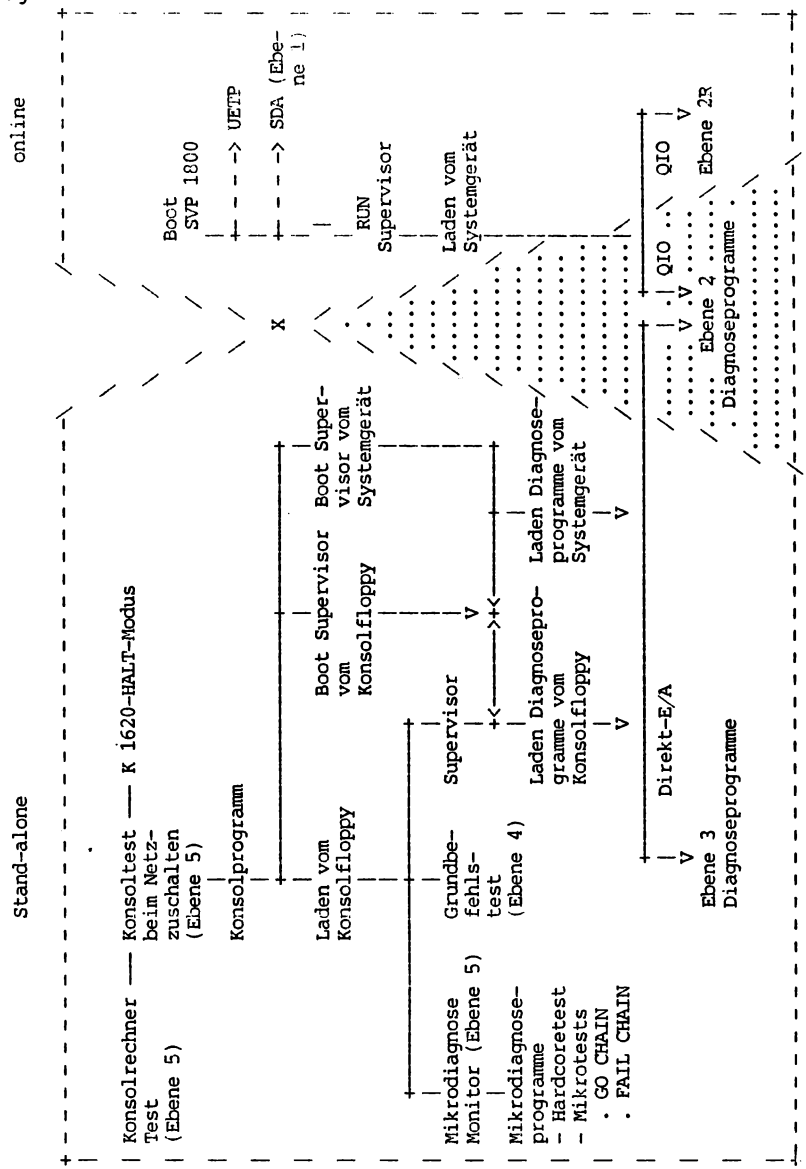

Bild 1-4: Laden von Diagnoseprogrammen und deren Steuerung

1.1.5. Diagnosestrategie

Eine große Anzahl von Diagnoseprogrammen, die in eine Hierarchie eingeordnet sind, bietet dem Anwender eine hohe Flexibilität in der Methode der Fehlerisolation. Dafür sind jedoch detaillierte Kenntnisse des Rechnersystems und des Diagnosesystems erforderlich.
Bei Systemfehlern sollte man, wenn möglich, die Online-Hilfsmittel nutzen; zuerst die Analyse der ERRLOG-Datei mittels System Dump Analyzer(SDA), aufgerufen durch ANALYZE/ERROR_LOG. Die Datei ERRLOG.SYS der Directory [SYSERR] sammelt bei Fehler signifikante Informationen über den gegenwärtigen Stand des Systems.
Der System Dump Analyzer, aufgerufen durch ANALYZE/CRASH_DUMP, hilft, SVP-Crash (Zusammenbruch des Betriebssystems) zu analysieren.
UETP dient der Prüfung des Zusammenspiels von Hard- und Software.
Die Online-Diagnoseprogramme identifizieren das fehlerhafte Subsystem. Wenn dieses bekannt ist, sollten die Diagnoseprogramme der Ebenen 2 und 3 angewendet werden. Ist es jedoch nicht möglich, SVP zu starten oder den Diagnose-Supervisor vom Systemgerät zu booten, muß man versuchen, den Supervisor vom Konsolfloppy zu booten oder zu laden, um danach die Diagnoseprogramme zu laden. Sollte dies auch nicht möglich sein, muß versucht werden, die Mikrodiagnoseprogramme oder die Programme der Ebene 4 abzuarbeiten.

1.1.6. Kodierungskonvention für Diagnoseprogramme

Die Diagnoseprogramme werden durch fünf alphanumerische Zeichen, gefolgt von der Revisionsnummer, bezeichnet. Die Kodierungskonvention ist nach einer Baumstruktur aufgebaut. Damit ist die Kodierung leicht einprägbar und ausbaufähig.

```
     P  R  G  T  V          A.B
     ____________________________

P = Prüfsystemunterlage für Rechner mit virtuellem Speicher (RVS)
___
 |
 |
 |
R = RVS-Rechnerreihe K 1800
___
 |    S = System K 1840
 |    V = variabel (gilt für alle Systeme der Rechnerreihe K 1800)
 |
G = Gerätekategorie  T = Gerätetyp
___                  ___
 |    A = Anwendung
 |    |   AA = Zeilendrucker LP
 |    |
      |
      C = Kanal
      |   CA = MSBUS-Adapter
      |   CB = SKRBUS-Adapter
      |
      D = Kommunikationsgeräte
      |   DA = AMF18 (Multifunktions-Kommunikations-Kontroller)
      |   DL = AMF18        "              "            "
      |   DR = AR16W (DMA-Parallel-Kontroller)
      |   DW = RONAS (Netzkontroller)
```

```
|
K = CPU
|
|
|
L = Datenbasisfiles
|   LB = Systemexerciser Datanbasisfile
|
|
M = Magnetband
|   MA = CM 5307
|   MZ = CM 5306, CM 5308, CM 5309
|
Q = QIO Gerätedriver
|   Für diese Kategorie stellen der vierte und fünfte Buchstabe des
|   Kodes das Gerätemnemonik dar.
|   z.B. PVQDB ist der QIO Gerätedriver für CM 5404/CM 5416
|
R = Rotierende Massenspeicher
|   RA = mehrere Gerätetypen können getestet werden
|   RB = CM 5404 / CM 5416
|   RC = Kontrollertest für CM 5404 / CM 5416
|   RD = K 5502 - Test
|   RG = K 5502 - Test
|
S = System-Monitor
|   SA = Supervisor
|   SB = Autosizer
|
T = Terminals
|   TA = alle Terminals
|   TB = Terminal Exerciser
|   TS = Seriendrucker
|   TX = IGT (interaktives grafisches Terminal)
|   TY = Plotter
|   TZ = Digitalisiergerät
|
U = Hilfsprogramme
|   UA = Script-File
|   UB = Kopier-Kommando-File
|
X = Systemexerciser
|   XB = System oder Businteraction
|
|
```

V = Das fünfte Zeichen (A bis Z und 0 bis 9) dient der weiteren Spezifizierung
—— Beispiel: PVTX1 ist das Diagnoseprogramm für das Gerät IGT1
PVTX2 " " " " " " IGT2

A = Hauptrevisionsnummer
——

B = Unterrevisionsnummer
——

Die Diagnoseprogramme zum Test des Konsolrechners, seiner peripheren Geräte und einige andere Dateien der Ebene 5 gehorchen nicht dieser Konvention.

1.1.7. Die Diagnosedirectory SYSMAINT und deren Pflege

Die Directory SYSMAINT ist Bestandteil des Betriebssystems SVP 1800 und enthält die Diagnoseprogramme, andere Dateien für Diagnose-, Wartungszwecke und Systemanlauf und den Diagnoseindex AAPNX.LIS (Textdatei). AAPNX.LIS definiert eindeutig den Inhalt der jeweiligen Ausgabe und wichtige Hinweise zur Nutzung der SYSMAINT. Die Diagnoseprogramme zum Test des Konsolrechners und seiner peripheren Geräte sind nicht Bestandteile der Directory SYSMAINT. Parallel dazu werden die Komponenten der Directory auf Diskette ausgeliefert (außer Diagnoseprogrammen der Ebene 2R). Damit hat der Anwender bei Defekten des E/A-Subsystems die Möglichkeit, die Fehlersuche mittels der auf der Diskette gespeicherten Programme durchzuführen. Siehe Punkt 1.1.4. (Laden der Diagnoseprogramme und deren Steuerung).
Defekte Disketten können durch neue, aus der SYSMAINT kopierte Disketten ersetzt werden. An den Anwender werden für Systemerweiterung erforderliche Diagnoseprogramme im allgemeinen auf Diskette nachgeliefert. Auch diese Programme sollten in der SYSMAINT deponiert werden.
Das SVP-Hilfsprogramm EXCHANGE.EXE dient für beide Fälle als Hilfsmittel. Weiterhin wird zur Unterstützung des Aufbaues von Disketten die Kommandodatei DSCOPY.COM der Directory SYSMAINT empfohlen. Der Inhalt der Directory kann im User-Modus über das SCL-Kommando DIR angezeigt werden.

```
$ DIR XX:[SYS0.SYSMAINT]
```

XX = logischer Name des Systemdatenträgers

Die SYSMAINT enthält folgende Dateigruppen:
- o alle Diagnoseprogramme der Ebenen 2R bis 5
- o alle HELP-Dateien, die den Diagnoseprogrammen zugeordnet sind
  Die Kennzeichnung ist Programm-Name.HLP.
- o alle Offline-Driver, auf die der Supervisor im Offline-Modus der Testebene 2 zurückgreift (Bezeichnung siehe Pkt. Diagnoseprogramm-Kodierungskonvention)
- o alle BOOT-Files, die zum Booten des Betriebssystems SVP 1800 und zum Booten des Supervisors von den verschiedenen Datenträgern erforderlich sind
- o alle Hilfsdateien, die für den Anlauf des Systems nach dem Einschalten des Netzes erforderlich sind, einschließlich weiterer HELP-Dateien
  Beispiele sind:
  Konsolprogramm (CONSOL.SYS)
  Konsolprogramm-HELP (CONSOL.HLP)
  Primär-Bootstrap Programm (VMB.EXE)

1.2. Dateien der Konsolebene und deren Aufgabe

1.2.1. Übersicht über die Dateien und die zugeordneten Disketten

Zu den Dateien der Konsolebene gehören nicht nur die lauffähigen Diagnoseprogramme und deren Monitore (Dateityp .EXE, .SYS, .TSK), sondern auch Helpdateien (.HLP) und Kommandodateien (.CMD).
Die Dateien der Konsolebene sind auf fünf Disketten untergebracht und können vom Konsolladegerät geladen/abgearbeitet werden.

Disketteninhalt:

DS0: Konsolsubsystemtests

(Tests des K 1620, Anschlußsteuerungen, Floppy und Terminal unter Steuerung des LPROEM, siehe Kapitel 2 des Handbuches)

DS1: K 1840 Standardkonsole

- Abarbeitungsfähige Dateien:

CONSOL.SYS	Konsolprogramm
VMB.EXE	Primär Bootstrap-Programm

- Helpdateien

CONSOL.HLP	Helpdatei zum Konsolprogramm
ABBREV.HLP	" für Abkürzungen der Konsolkommandosprache
ERROR.HLP	" Fehlererläuterung bei Konsoloperationen
BOOT.HLP	Bootstrap-Help für VMB.EXE

- Kommandodateien für SVP-BOOT und Supervisor-BOOT von verschiedenen Medien

Konsolfloppy

CS1BOO.CMD	Boot-Kommandodatei für Stand-alone Backup des SVP von Konsoldiskette
CS1GEN.	Bootkommandodatei für Stand-alone Backup des SVP von Konsoldiskette mit Systemparametereingabe

CM 5404 / CM 5416 SVP 1800-BOOT

DB0BOO.CMD	DB0-Boot (Gerät 0)
DB0BOO.ILV	DB0-Boot und überlappter Speicherbetrieb
DB0GEN.	" mit Systemparametereingabe
DB0XDT.	" " " und DEBUG
DB1BOO.CMD	DB1-Boot (Gerät 1)
DB1GEN.	DB1-Boot mit Systemparametereingabe

DB7BOO.CMD	DB7-Boot (Gerät 7)
DB7GEN.	DB7-Boot mit Systemparametereingabe
DBABOO.CMD	DB-Boot mit Eingabeforderung für Geräteeinheit vor Start der Kommandodatei (MSBUS auf TR = 8)
DBAGEN.	DB-Boot mit Eingabeforderung für Geräteeinheit vor Start und Systemparametereingabe nach Start der Kommandodatei
DBAXDT.	DB-Boot wie DBAGEN mit DEBUG

DBBBOO.CMD wie DBABOO.CMD (MSBUS auf TR = 9)
DBBGEN.
DBBXDT.

CM 5404 / CM 5416 für Diagnose-Supervisor-Boot

SBOBOO.CMD DB0-Supervisor Boot-Kommandodatei
SB1BOO.CMD DB1-Supervisor Boot-Kommandodatei

.

SB7BOO.CMD DB7-Supervisor Boot-Kommandodatei
SBABOO.CMD DBA-Supervisor Boot-Kommandodatei mit Eingabeanforderung für Geräteeinheit vor Start und Parametereingabe nach Start der Kommandodatei
SBBB00.CMD wie SBA00.CMD (MSBUS auf TR9)

K 5502 für SVP 1800-Boot

Da der Gerätetyp für K 5502 im SVP 1800 durch DR kodiert ist, ist die Bezeichnung der Boot-Kommandodateien für K 5502 analog der CM 5404/CM 5416.
Beispiel:
DR0BOO.CMD DR0-Boot-Kommandodatei für K 5502
Weiterhin kann eine Standard-Boot-Kommandodatei DEFBOO.CMD für das Standardgerät als Systemträger für eines der Plattensysteme definiert werden.

K 5502 für Supervisor

SR0BOO.CMD DR0-Supervisor Boot-Kommandodatei
SR1BOO.CMD DR1-Supervisor Boot-Kommandodatei

.

SR7BOO.CMD DR7-Supervisor Boot-Kommandodatei
SRABOO.CMD DRA-Supervisor Boot-Kommandodatei mit Eingabeanforderung für Geräteeinheit vor Start und Parametereingabe nach Start der Kommandodatei
SRBB00.CMD wie SRAB00.CMD (MSBUS auf TR9)

- Weitere Kommandodateien

CRASH. Kommandodatei bei SVP-Absturz
RESTAR.CMD Kommandodatei für RESTART
RESTAR.ILV " " " für überlappte Speicherarbeit

- Datendateien

WCS124.PAT Speicherzellen 1000 - 17FF des ladbaren Mikroprogrammspeichers

DS2: K 1840 MIKRODIA #1

PSKAB.SYS (MICMON.SYS)	Mikrodiagnosenmonitor
PSKAC.TSK	Hardcore Monitor
PSKAD.TSK	Hardcore Test
PSKAE.TSK	Mikrotestmonitor
PSKAF.TSK	Kommandoeingabeprogramm
PSKAG.TSK	Directory-Programm
PSKAK.TSK	Mikrotestfehlermonitor
PSKAH.TSK	Mikrotest 1
PSKAL.TSK	Mikrotestfehler 1

DS3: K 1840 MIKRODIA #2

PSKAB.SYS (MICMON.SYS)	Mikrodiagnosenmonitor
PSKAE.TSK	Mikrotestmonitor
PSKAF.TSK	Kommandoeingabeprogramm
PSKAG.TSK	Directoryprogramm
PSKAK.TSK	Mikrotestfehlermonitor
PSKAR.TSK	Mikrotest 2
PSKAS.TSK	Mikrotestfehler 2

DS4: Reserviert für Ergänzungen

1.2.2. Diskette DS0 (Konsolsubsystemtest)

Die Diskette DS0 enthält ein für das Konsolsubsystem (KSS), bestehend aus Rechner K 1620, Anschluß-Steuerung für Folienspeichereinheit (AFS) und Anschluß-Steuerung für Konsolterminal K 8911.80 (AIS) selbständiges Diagnosesystem.
Den Kern des Diagnosesystems bildet eine Modifikation des Leitprogrammes LPROES der Rechnerfamilie K 1600, genannt LPROEM. Der Start des LPROEM kann auf zwei Wegen erfolgen:

1. Automatisches Booten des LPROEM
   1. AUTORESTART in Stellung "AUS" (am Rechnerschrank)
   2. Einlegen der Diskette DS0 in die Folienspeichereinheit
   3. Einschalten des Netzes
   4. Schlüsselschalter in Stellung LOCAL (am Rechnerschrank)
   5. Danach wird automatisch der sich auf dem ROM des Konsolinterfaceboards (CIB) befindliche Boot-Lader gestartet, der die Anschluß-Steuerungen des K 1620 initialisiert und einige K 1620-Tests abarbeitet. Diese Tests bilden die Voraussetzung für die weitere Arbeit mit dem Konsolsubsystem.
      Danach erfolgt das Laden des BOOT-Blockes zum Laden des gesamten LPROEM und dessen Start. Der BOOT-Block ist Bestandteil des LPROEM und erster Block der Diskette.
      Die Beschreibung des LPROEM und die dem Konsolsubsystem unter LPROEM zugeordneten Tests erfolgen in einem gesonderten Kapitel.
2. Booten des LPROEM aus dem K 1620-HALT-Modus
   Der K 1620-HALT-Modus ist gekennzeichnet durch die folgende Ausgabe auf dem Konsolterminal.

   nnnnnn
   @

nnnnnn Adressen des nächsten auszuführenden Befehls
@ fordert eine Bedienereingabe

Der BOOT-Lader für die Folienspeichereinheit (AFS) kann aus dem K 1620-HALT-Modus folgendermaßen angesprungen werden:

- Drücken der Taste "Reset" am KSS
- @173000G
- @P, wenn R7 des KSS schon auf 173000 steht

In den beiden zuletzt genannten Fällen ist darauf zu achten, daß der Stack des KSS den Wert 400 hat (@R6/000370 400), bevor das Kommando gegeben wird. Wird der BOOT-Lader nach einer der drei genannten Varianten (die erste ist die bequemste) gestartet, wird der Bootblock der eingelegten Diskette (Sektor 1, Spur 1) gelesen und dann die Regie dem gebooteten Programm (Absolutlader) übergeben. Es können also alle Disketten gebootet werden, die auf Sektor 1, Spur 1 einen Absolutlader enthalten.
Liegt die Diskette DS0 in der Konsole, wird zuerst der Bootblock der Diskette DS0 und bei dessen Abarbeitung das gesamte LPROEM in den Hauptspeicher des K 1620 geladen und gestartet.
Der K 1620-HALT-Modus wird erreicht durch:

o Abarbeitung eines HALT-Befehles
o Bei Netzeinschalten oder Drücken der RESET-Taste des Konsolrechners bei gedrückter HALT-Taste meldet sich das Konsolterminal mit

173000
@

o Drücken der Unterbrechungstaste (BREAK) am Konsolterminal
o bei Doppelbusfehler
o bei Konsolprogramm-Absturz (siehe 1.2.3.2.)

1.2.3. Diskette DS1 (RVS K 1840 Standardkonsole)

1.2.3.1. Aufgaben der Standardkonsoldiskette

- Die Diskette DS1 ist für jeden Systemanlauf erforderlich
- Booten des Konsolprogrammes und Laden des WCS
- Booten des SVP 1800 von verschiedenen Datenträgern
- Booten des Diagnose-Supervisors von verschiedenen Datenträgern
- Ermöglichen des Restarts des SVP 1800 aus dem Konsol-E/A-Modus
- Start der CRASH-Kommandodatei vom Konsol-E/A-Modus nach Systemabsturz

1.2.3.2. Booten des Konsolprogramms

Dafür ist folgende Prozedur möglich:

- Booten bei Netzeinschaltung
  o Einschieben der Diskette DS1 in die Folienspeichereinheit
  o AUTORESTART-Schalter in "AUS"-Stellung, damit SVP 1800 nicht gebootet wird und die Konsole im Konsol-E/A-Modus bleibt
  o Schlüsselschalter in Stellung "LOCAL"

- o Netzzuschaltung
  Mit der Netzeinschaltfolge wird das Konsolbootstrap-Programm auf dem Konsolinterfaceboard gestartet. Danach werden Initialisierungsfunktionen für den K 1620 und K 1620-Tests abgearbeitet.
- o Automatischer Start und Abarbeitung des Konsolbootstrap-Programmes, das sich auf einem ROM des Konsolinterfaceboards des Rechners K 1840 befindet.
- o Das Bootstrap-Programm initialisiert die Anschluß-Steuerungen AFS des K 1620, testet den K 1620 in seinen Grundfunktionen, lädt den Bootblock des CONSOL.SYS von der Diskette und startet ihn.
- o Abarbeitung des Bootblockes führt zum Laden des Gesamtprogrammes CONSOL.SYS, zur Initialisierung der CPU K 1840 und unter anderen Funktionen auch zum Laden des WCS.

- Boot aus dem K 1620-HALT-Modus
  - o Eingabe:

    @140200G

  - o Im Falle eines Konsolprogramm-CRASH während laufendem Betriebssystem ist die Eingabe

    @141330G

    möglich, um das Konsolprogramm neu zu booten, ohne das laufende Betriebssystem zu beeinflussen.
  - o Mit >>> fordert das Konsolprogramm am Konsolterminal Eingaben vom Operator an. Das Konsolprogramm befindet sich jetzt im Konsol-E/A-Modus. Die Beschreibung der Kommandos für die Konsolsprache kann jetzt durch die Eingabe von

    >>> HELP

    am Konsolterminal angezeigt oder aus dem Kapitel Konsolsteuerung entnommen werden.

1.2.3.3. Booten des SVP 1800 von verschiedenen Datenträgern

Das Booten des SVP 1800 ist auf zwei Wegen möglich.

- Booten, ausgehend vom Konsol-E/A-Modus
- Automatisches Booten des SVP 1800, ausgelöst durch Netzeinschaltung bei Schalterstellung "AUTORESTART EIN" oder durch Drücken des Tastenschalters BOOT am Rechnerschrank.

1.2.3.3.1. Booten des SVP 1800 vom Konsol-E/A-Modus

Man unterscheidet zwei Bootkommandodateisätze. Ein Bootkommandodateisatz arbeitet mit festen Systemparametern, der andere Satz arbeitet mit Wechsel der Systemparameter.
Zur Verfügung stehen die in 1.2.1. genannten Bootkommandodateien für die Plattenspeichersubsysteme CM 5404, CM 5416 und K 5502.

- Booten des Systems vom Standardgerät

  >>> BOOT

  Mit BOOT wird die Standardbootkommandodatei DEFBOO.CMD abgearbeitet. Die DEFBOO.CMD kann eine der Kommandodateien DBXBOO.CMD oder DRXBOO.CMD sein, jenachdem, welches Gerät DBX oder DRX das Standardgerät für das SVP 1800 sein soll.

- Booten des Systems von einem der Plattenspeicher DBX oder DRX ohne Systemparametereingabe
  X = 0...7 (Geräteeinheitennummer)

  >>> BOOT <gerätename>

  Beispiel:

  >>> BOOT DB1

  Mit diesem Konsolkommando wird die Kommandodatei DB1BOO.CMD gestartet und das SVP 1800 vom Plattenspeicher CM 5404 mit der Geräteeinheitennummer 1 gestartet.
  Analog dazu können die Bootkommandodateien DB0BOO.CMD ... DB7BOO.CMD und DR0BOO.CMD ... DR7BOO.CMD genutzt werden. Das Booten ist aber auch wie folgt möglich:

  >>> @<BOOT-Kommandodatei>

  Beispiel:

  >>> @DB1BOO.CMD

  Dieser Satz von Bootkommandodateien setzt einen Standard von Systemparametern voraus (z.B. Anschluß des Speichersubsystems an eine feste Systembusadreß-TR-Nummer, feste Blocknummer, wo sich der Bootblock befindet, bei SKR-Busanschluß feste Adresse des SKR-Bus-Steuerregisters, bei MSBUS-Anschluß feste Kontroller/Formatter-Nummer).

- Booten des Systems von einem Plattenspeicher DBX oder DRX mit Systemparametereingabe.
  Der Bootkommandodateisatz

  DB0GEN ... DB7GEN
  DR0GEN ...DR7GEN

  ermöglicht das Booten des Systems von einem Plattenspeicher, der nicht mit den Standardsystemparametern des Bootkommandodateisatzes

  DB0BOO.CMD ... DB7BOO.CMD
  DR0BOO.CMD ... DR7BOO.CMD

  erreichbar ist.
  Der Start dieses Kommandodateisatzes erfolgt durch

  >>> @<Bootkommandodateiname>

  Beispiel:

```
>>> @DB0GEN
```

Nach dem Start der Kommandodatei werden die Systemparameter in die Register R0 ...R5 eingetragen und das Primärbootstrapprogramm PSOAD (ROM-Programm) auf Adresse 20003000 gestartet.
Danach erfolgt der Start des Primärladers VMB.EXE der Konsoldiskette DS1. VMB.EXE übernimmt die Parameter der Bootkommandodatei, lädt den Sekundärlader [SYSEXE]SYSBOOT.EXE und überträgt die Steuerung an ihn zum Laden und Start des SVP 1800. Nähere Ausführungen sind der Bootstrap-Help-Datei BOOT.HLP der Diskette DS1, dem Kapitel 3 oder der Dokumentation für SVP 1800 zu entnehmen. Der Aufruf des BOOT.HLP erfolgt durch

```
>>> @BOOT.HLP
```

1.2.3.3.2. Booten des SVP 1800 durch Netzzuschaltung oder durch Boottaste

Voraussetzungen:

- Schalter "AUTORESTART EIN", Schlüsselschalter in Stellung "LOCAL"
- RVS K 1840-Standardkonsole, DS1 eingelegt
- Systemdatenträger muß auf der zu der Bootkommandodatei DEFBOO.CMD zugeordneten Plattenspeichereinheit aufgelegt sein.

Nach dem Netzeinschalten wird SVP 1800 gebootet. Ist das Netz bereits zugeschaltet, kann über die Boottaste des Rechnerschrankes das System erneut gebootet werden.

1.2.3.4. Booten des Diagnose-Supervisors von verschiedenen Datenträgern (Stand-alone)

- Automatisches Booten vom Systemplattenspeicher im Konsol-E/A-Modus:

Der Datenkommandodateisatz, bestehend aus

SB0BOO.CMD ... SB7BOO.CMD (für CM 5404 / CM5416)
SR0BOO.CMD ... SR7BOO.CMD (für K 5502)

dient dem automatischen Booten des Diagnose-Supervisors von den Geräten 0...7 der beiden Plattenspeichersysteme aus dem Konsol-E/A-Modus heraus. Voraussetzung für das erfolgreiche Booten ist, daß sich der Supervisor in der Directory SYSMAINT, die sich auf dem Systemplattenstapel befindet und der Sekundärlader DIAGBOOT.EXE ist (Standard).
Das Booten erfolgt durch Start des Primärladers VMB.EXE, der wiederum den Sekundärbootlader DIAGBOOT.EXE der Directory [SYSMAINT] startet, DIAGBOOT.EXE lädt und den Supervisor startet.

Beispiel:

```
>>> BOOT SB0
oder
>>> @SB0BOO.CMD
```

Der Supervisor wird vom Plattenspeichersubsystem CM 5404 bzw. CM 5416 der Geräteeinheit 0 gebootet.

- Interaktives Booten vom Plattenspeicher im Konsol-E/A-Modus:

Die Bootkommandodateien

```
SBABOO.CMD
SRABOO.CMD
```

dienen dem Booten des Diagnose-Supervisors für den Fall, daß die Directory des Supervisors und bzw. oder der Sekundärbootlader nicht dem Standard entsprechen.
Der Operator wird nach dem Start der Kommandodatei zweimal zur Eingabe aufgefordert, einmal zur Eingabe des Bootfilenamens (z.B. DIAGBOOT.EXE) und einmal zur Eingabe des Files, der zu booten ist (PSSAA.EXE).
Vor dem Start der Kommandodatei ist in R3 die Nummer der Geräteeinheit, von der zu booten ist, anzugeben.

Beispiel:

```
>>> DEPOSIT R3 0
>>> BOOT SBA
      .
      .
Enter name of bootfile: [SYSMAINT]DIAGBOOT.EXE
Enter name of file: [SYSMAINT]PSSAA.EXE


DS>
```

- Booten des Supervisors von der Diskette der im Konsol-E/A-Modus

Wie in 1.1.7. ausgeführt, befinden sich die Diagnoseprogramme sowohl in der Directory SYSMAINT als auch auf Disketten.
Zum Booten des Supervisors von der Diskette ist die Diskette DS8 (siehe 1.4.3.2.) erforderlich, die die Kommandodatei DEFBOO.CMD enthält. DEFBOO.CMD der DS8 ist verschieden vom DEFBOO.CMD der DS1.

1.2.3.5. Restart des SVP 1800 vom Konsol-E/A-Modus

Für den Restart des SVP stehen die Kommandodateien RESTAR.CMD und RESTAR.ILV zur Verfügung.
Manuell kann der Restart erfolgen durch:

```
>>> @RESTAR.CMD
```

Der automatische Restart des Systems nach Netzausfall setzt voraus, daß der AUTO-RESTART-Schalter gesetzt ist. Sind zwei Speicherkontroller vorhanden, die überlappt arbeiten sollen, sollte RESTAR.CMD durch RESTAR.ILV ersetzt werden. RESTAR.ILV erhält dann den Namen RESTAR.CMD.

1.2.3.6. CRASH-Kommandodatei

Die CRASH-Kommandodatei CRASH. sollte nur genutzt werden:

- bei Systemabsturz in den Konsol-E/A-Modus,
- wenn SVP keine Kommandoeingabe akzeptiert, d.h. auch OPCCRASH nicht startbar ist
- wenn OPCCRASH Fehler bringt

```
>>> @CRASH
```

Die angezeigten Register sollen dem Operator helfen, den Systemabsturz zu analysieren. Im Normalfall sollte immer SHUTDOWN.COM gestartet werden. Ist SHUTDOWN.COM nicht ausführbar, sollte zunächst OPCCRASH gestartet werden.

Mögliche Eingaben:

```
$ @SYS$SYSTEM: SHUTDOWN          ;Start von SHUTDOWN für norma-
                                 ;len Abbruch des SVP

$ RUN SYS$SYSTEM: OPCCRASH       ;wenn SHUTDOWN nicht erfolgreich

>>> @CRASH                       ;wenn OPCCRASH nicht erfolgreich
```

1.2.4. Diskette DS2 (K 1840 MIKRODIA #1)
Diskette DS3 (K 1840 MIKRODIA #2)
Diskette DS4 (reserviert für Ergänzungen)

Diese Disketten enthalten die in 1.2.1. aufgeführten Diagnoseprogramme und deren Monitore.
Das Kapitel "Mikrodiagnose und Tests mittels WCS" beschreibt ausführlich die Monitorkommandos für die Abarbeitung der Mikrodiagnosen.
Ausgehend vom Konsol-E/A-Modus sind folgende Eingaben möglich:

```
>>> T
```
Start der Mikrodiagnosen und automatische Abarbeitung in steigender Reihenfolge

```
>>> T/COM
MIC>
```
Laden und Start des Mikrodiagnosemonitors, der mit MIC zur Eingabe von Monitorkommandos für die Bedienung der Mikrodiagnoseprogramme auffordert

1.3. Dateien der Diagnoseebene 4

Die Diagnoseprogramm PVKAA.EXE und PVKAS.EXE sind die Programme der Diagnoseebene 4. Sie sind in MACRO 18 geschrieben und arbeiten ohne Steuerung des Diagnose-Supervisors stand-alone. Die erfolgreiche Abarbeitung von PVKAA ist die Voraussetzung für den Start des Supervisors.
PVKAA.EXE befindet sich gemeinsam mit den Architekturbefehlstests PVKAB.EXE und PVKAS.EXE auf der Diskette DS 5.
Ausgehend vom Konsol-E/A-Modus wird PVKAA.EXE ab der physischen Adresse 0 geladen. Die Startadresse ist 200.

```
>>> LOAD PVKAA.EXE
>>> START 200
```

Analog dazu wird PVKAS.EXE gestartet. Eine eingehende Beschreibung von PVKAA.EXE erfolgt im Kapitel "PVKAA Grundbefehlstest" und für PVKAS.EXE im Kapitel "PVKAS Grundspeichertest".

1.4. Dateien der Diagnoseebenen 3 und 2

1.4.1. Übersicht

Zu den Dateien der Diagnoseebenen 3 und 2 gehören der Diagnose-Supervisor PSSAA.EXE als zentraler Steuermonitor für alle Diagnoseprogramme (Typ.EXE) der Diagnoseebenen 3, 2 und 2R, die Helpdatei des Supervisors (PVSAA.HLP), die Helpdateien der Diagnoseprogramme und Kommandodateien für den Supervisor. Diese Dateien sind, wie in 1.1.7. aufgeführt, Bestandteil der Diagnosedirectory SYSMAINT des Betriebssystems SVP 1800 und sind gleichzeitig auf Diskette für offline-Tests (Stand-alone) abgelegt).
Wie in 1.1.4., Bild 1-4, aufgezeigt, ist der Supervisor vom Konsol-E/A-Modus ausgehend startbar durch

- o Laden und Start von Diskette
- o Booten von Diskette
- o Booten vom Systemgerät (Plattenspeicher)

Nach dem Start des Supervisors sind die Diagnoseprogramme unter Supervisor sowohl von der Diskette als auch vom Systemgerät ladbar und startbar. Der Start der Diagnoseprogramme unter online-Bedingungen kann nur durch Start des Supervisors unter SVP 1800 erfolgen. Der Diagnose-Supervisor ist also sowohl standalone als auch online lauffähig.
Die vom Supervisor unterstützten Diagnoseprogramme der Ebene 3 liefern ihr eigenes Geräteinterface (direkte E/A) und können deshalb nur stand-alone abgearbeitet werden.
Die Diagnoseprogramme der Ebene 2 dagegen können stand-alone und online arbeiten. Der stand-alone-Modus ist deswegen möglich, weil der Supervisor die E/A-Servicedienste (QIO-Service) liefert. Der Supervisor emuliert in diesem Falle das SVP 1800. Die dafür notwendigen Stand-alone-Driver sind nicht Bestandteil des Supervisors. Sie werden vom Supervisor bei E/A-Anforderung vom Speichermedium (Diskette oder Plattenspeicher) geladen. Sie sind nicht mit dem SVP-Driver identisch.
Sollen die Diagnoseprogramme, die QIO-Service fordern, unter SVP 1800 arbeiten, übergibt der Supervisor die QIO-Anforderungen direkt dem SVP 1800.

1.4.2. Disketten

Die Disketten DS0 ... DS4 sind der Konsolebene zugeordnet.
Die Disketten ab DS5 sind für die Programme der Diagnoseebenen 3 und 2 vorgesehen (außer PVKAA.EXE und PVKAS.EXE. Die Diskette DS8 als Träger des Diagnose-Supervisors nimmt dabei eine zentrale Stellung ein.
Für die Diagnoseebene 2R werden die Programme NICHT auf der Diskette bereitgestellt, da diese Programme einschließlich Supervisor von der Directory SYSMAINT geladen werden.

Zusammenstellung der Disketten:

DS5:	BEFEH #1	) Befehls-
DS6:	BEFEH #2	) tests
DS7:	BUS + AUTOKONFIG DIAG	Busadaptertest und Autokonfiguration
DS8:	K 1840 DIAG SUPER	Diagnose-Supervisor
DS9:	K 1840 DISK DIAG #1	) Platten-
DS10:	K 1840 DISK DIAG #2	) speicher-
DS11:	K 1840 DISK DIAG #3	) tests
DS12:	AMF18 K 8081 DIAG	AMF18 Tests
DS13:	AR16W K 6081 DIAG	AR16W Tests
DS14:	RONAS K 8681 DIAG	RONAS Tests
DS15:	K 1840 TAPE DIAG	Magnetbandtests

Die den einzelnen Disketten zugeordneten Programme sind in einer Übersicht im Anhang zusammengefaßt. Eine genaue Beschreibung der Programme erfolgt in den entsprechenden Kapiteln des "Diagnosehandbuchs Rechner RVS K 1840" bzw. im "Diagnosehandbuch Peripherie RVS K 1840".

1.4.3. Laden und Start der Diagnoseprogramme der Ebenen 2 und 3 im Stand-alone-Modus

1.4.3.1. Laden und Start des Supervisors von der Diskette

- Im Konsol-E/A-Modus wird die Diskette DS1 durch die Diskette DS8 ersetzt.

  ```
  >>> LOAD PSSAA.EXE /START: FE00
  >>> START 10000
  ```

  Danach meldet sich der Diagnose-Supervisor mit DS> zur weiteren Kommandoeingabe.

- Im Supervisormodus ist es möglich, das abzuarbeitende Diagnoseprogramm sowohl von der Diskette (a) als auch vom Systemplattenspeicher (b) zu laden und zu starten.

  a) Ersetzen der Diskette DS8 durch die Diskette, die das auszuführende Diagnoseprogramm trägt. Kommandoeingabe gemäß Supervisorbeschreibung und Diagnoseprogrammbeschreibung

  b) Laden/Start des Diagnoseprogramms vom Systemgerät im Supervisormodus Siehe Beschreibung Diagnose-Supervisor und des Diagnoseprogrammes.

1.4.3.2. Booten des Supervisors von der Diskette

Im Konsol-E/A-Modus wird die Diskette DS1 durch die Diskette DS8 ersetzt.

```
>>> BOOT
```

Boot startet die Kommandodatei DEFBOO.CMD der Diskette DS8. DEFBOO.CMD lädt und startet PSSAA.EXE von Diskette DS8. Der Supervisor meldet sich mit der

Eingabeanforderung DS>.

1.4.3.3. Booten des Supervisors vom Systemgerät

Dies wurde bereits im Punkt 1.2.3.4 ausführlich behandelt.

1.4.4. Start der Diagnoseprogramme der Ebene 2 im online-Modus

Voraussetzung ist das lauffähige Betriebssystem SVP 1800. Nach Operatoranforderung durch $ kann der Supervisor gestartet werden.

```
$ RUN [SYS0.SYSMAINT]PSSAA
DS>
```

Für die Abarbeitung der Diagnoseprogramme im online-Modus werden bestimmte Systemprivilegien gefordert. Diese müssen vom Systemmanager dem Nutzer zugewiesen werden. Die erforderlichen Privilegien und die weitere Bedienung siehe Supervisorbeschreibung.

1.5. Dateien der Diagnoseebene 2R

Die Diagnoseprogramme dieser Ebene sind nur unter SVP 1800 bei Steuerung durch den Supervisor abarbeitbar. Der Start des Supervisors von der SYSMAINT erfolgt durch

```
$ RUN [SYS0.SYSMAINT]PSSAA
DS>
```

Die weitere Bedienung erfolgt gemäß Beschreibung des Diagnose-Supervisors und der Diagnoseprogrammbeschreibungen.

1.6. Dateien der Diagnoseebene 1

1.6.1. Allgemeines

Das Betriebssystem SVP 1800 liefert bezüglich Wartung der Gerätetechnik mehrere Hilfsmittel:

- UETP (User Environment Test Package)
  Dieses Programmpaket enthält eine Sammlung von Tests, die demonstrieren, ob Software und Hardware im Zusammenspiel richtig arbeiten. UETP ist aber kein Diagnoseprogramm.

- SDA (System Dump Analyzer)
  Der SDA ist ein Dienstprogramm zur Bestimmung der Ursache des Systemzusammenbruchs (ANALYZE/CRASH_DUMP), Systemfehlern (ANALYZE/ERROR_LOG) und zu

Untersuchungen am laufenden System (ANALYZE/SYSTEM).

1.6.2. User Environment Test Package (UETP)

UETP prüft das Gesamtsystem von Hardware und Software.
UETP testet nicht nur die Arbeit einzelner Komponenten der Hard- und Software, sondern auch die Komponenten in ihrer Zusammenarbeit als ein integriertes System. Dazu werden eine Reihe von Tests durchgeführt, die eine typische Nutzerumgebung simulieren.
UETP ist kein Diagnoseprogramm, d.h. es testet nicht jede und alle Systemeigenschaften erschöpfend. Wenn UETP ohne harten Fehler seine Arbeit beendet, ist das getestete System bereit für seine Nutzung durch den Anwender.
UETP prüft Geräte und Grundfunktionen des SVP 1800 mit Ausnahme der Compiler.
Die getesteten Systemkomponenten sind:

- Standardperipheriegeräte
- die Mehrnutzerfähigkeit
- die Netzfähigkeit

1.6.2.1. Einloggen und Vorbereitung der Geräte für den Test durch UETP

- Der Start des UETP erfordert einen gesonderten Account. Nach dem Ausloggen des Nutzers muß sich der Nutzer mit dem Namen SYSTEST dem System bekanntgeben.

```
$ LOGOUT

.
USERNAME: SYSTEST
PASSWORD: UETP
```

- Man sollte sich am Konsolterminal einloggen.
- Alle Dateien für das Programmpaket befinden sich in der Directory [SYSTEST].
- Vorbereitung der Geräte für den Test
  - o Plattenspeicher:
    Plattenspeicher müssen initialisiert sein, eine Directory [SYSTEST] enthalten und mit MOUNT/SYSTEM dem System bekannt sein. Beim Test werden keine Dateien auf dem Testgerät überschrieben.
  - o Magnetbandgeräte:
    Auf jedem zu testenden Magnetbandgerät sollte ein Testmagnetband aufgelegt sein. Es muß auf die Bandmarke positioniert und online geschaltet sein.
    Wenn nötig, ist das Magnetbandgerät zu inititialisieren. Mit MOUNT ist das Gerät dem System bekannt zu machen.
  - o Alle anderen Geräte müssen ebenfalls online geschaltet sein.
  - o UETP testet nicht folgende Geräte:
    Konsolterminal
    Konsolladegerät
    das Terminal, das den Test des UETP aufgerufen hat
    Terminals mit Wahlleitungsanschluß
    Nichtstandardgeräte

1.6.2.2. Start des UETP

Aufruf der Kommandoprozedur UETP.COM in der Directory [SYSTEST]

```
$ @UETP
```

UETP antwortet wie folgt:

```
Run "ALL" UETP phases or a "SUBSET"[ALL]?
```

ALL ist Standard. Nach <CR> antwortet UETP mit folgenden Fragen:

```
How many passes of UETP do you wish to run [1]?  <CR>
How many simulated user loads do you want [n]?   <CR>
Do you want long or short report format [long]?  <CR>
```

Die in eckigen Klammern angegebenen Größen sind Standard und können mit <CR> beantwortet werden oder es muß eine andere Eingabe erfolgen.
Erfolgt auf die erste Frage die Eingabe SUBSET, kann man eine oder mehrere der Phasen

```
Device, LOAD, NET, CLUSTER
```

auswählen. UETP fordert danach weitere Eingaben an. Die einzelnen Phasen sind in der Dokumentation für das SVP 1800 (Teil UETP) ausführlich erläutert.
Auf die zweite Frage kann die Anzahl der Durchläufe eingegeben werden.
Die Eingabe für die maximale Anzahl der simulierten Nutzer hängt von der Speichergröße und dem Swapping- und Paging-Raum des Systems ab.
UETP berechnet den Standardwert aus diesen Größen selbst. Den berechneten Standard kann man wie folgt erhalten:

```
$ DEFINE MODE DUMP
$ RUN UETINIT00
```

UETINIT00 dient nicht zum Einleiten einer Testphase, sondern nur dazu, die Gleichung zur Bestimmung der maximalen Anzahl der Nutzer und der speziellen Faktoren (z.B. Anzahl der Pages, modifizierten Pages, Größe des Workingset) anzuzeigen.
Wird bei der dritten Frage "Langes Format" ausgewählt, sendet UETP sowohl alle Fehlernachrichten als auch Informationen über den Beginn und das Ende jeder Phase und jedes Tests zum Konsolterminal. Wird "Kurzes Format" ausgewählt, zeigt UETP Statusinformationen, Fehlernachrichten und den Beginn und das Ende einer jeden Phase an.
Da UETP alle seine Ausgaben in die Datei UETP.LOG schreibt, kann bei Fehlern nach weiteren Informationen im UETP.LOG gesucht werden.

1.6.2.3. Unterbrechung des UETP

CTRL/Y bricht UETP während des Laufs der Kommandoprozedur ab. CTRL/Y unterbricht einen laufenden UETP-Test und die Steuerung kehrt zum Kommandointerpreter zurück. Es können weitere Kommandos eingegeben werden (z.B. CONTINUE, STOP, EXIT).
CTRL/C unterbricht die Kommandoprozedur. UETP geht automatisch zur nächsten Phase in der Kommandoprozedur über.

1.6.2.4. Fehler

- UETP reagiert bei Fehler wie ein typisches Nutzerprogramm. Es gibt entweder eine Fehlernachricht aus und setzt seine Arbeit fort, oder schließt das Programm/Phase bei einem Abbruchfehler ab. UETP versucht aber nicht, die Ursache des Fehlers zu ergründen.
- Das Terminal, von dem UETP aufgerufen wurde, zeigt immer die Statusinformationen, wie z.B. Nachrichten über Beginn und Ende einer jeden Phase, und Fehlernachrichten an. Ausführliche Nachrichten über die Tests werden in den Log-Dateien abgelegt. Es ist sinnvoll, diese Log-Dateien zu prüfen, selbst dann, wenn keine Fehler auf dem Terminal angezeigt wurden.

1.6.2.5. Tests

- Die Kommandoprozedur UETP.COM beinhaltet Kommandos, die jede Testphase einleiten (automatischer Test).
- Die Gerätephase beinhaltet separate Tests für jeden Gerätetyp.
- Durch Auswahl von Testimages für die verschiedenen Geräte können Einzeltests durchgeführt werden, z.B.:

UETDISK00.EXE	Plattenspeichertests
UETTAPE00.EXE	Magnetbandtests
UETTTYS00.EXE	Terminals und Drucker

```
$ RUN UETTTYS00.EXE
Controller designation?: TTB
Start des Terminaltests
```

- Die Tests können im Schleifenmodus abgearbeitet werden.

```
$ DEFINE MODE LOOP
```

- Funktionen der Tests:
  - o Für jeden eigegliederten Plattenspeicher schafft der Test zwei Dateien, in die Blöcke von Daten geschrieben werden. Der Test prüft die Dateien, meldet jeden Fehler über SYS$OUTPUT und löscht diese Plattendateien bei Testende.
  - o Der Magnetbandtest prüft alle Magnetbandgeräte des Systems. Nach dem Test wird das Magnetband zurückgespult und neu initialisiert.
  - o Der Terminal- und Druckertest generiert ein Testmuster.
- Der System-Lade-Test:
  Er simuliert eine Anzahl von Terminalnutzern, die simultan Systemressourcen anfordern.

1.6.2.6. Abschluß des UETP

Am Ende des UETP-Durchlaufes zeigt UETP die Uhrzeit an und bestimmt, ob es erneut gestartet werden muß (mehrere Durchläufe). Am Ende des gesamten UETP-Laufes werden die temporären Dateien gelöscht.

1.6.3. System Dump Analyzer (SDA)

1.6.3.1. Aufruf

Der SDA befindet sich als SDA.EXE in der Directory [SYSEXE]. SDA wird wie folgt aufgerufen:

```
$ ANALYZE/CRASH_DUMP  Dateispezifikation zur Analyse eines Crash
$ ANALYZE/SYSTEM      zur Analyse des laufenden Systems
$ ANALYZE/ERROR_LOG   zur Analyse des Error-Log-Files
```

Es ist sinnvoll, daß SDA privilegiert installiert wird, damit alle Nutzer mit unzureichenden Privilegien SDA nutzen können.

1.6.3.2. ANALYZE/CRASH_DUMP

Bei Crash schreibt die Kernelroutine den Inhalt des Error-Log-File-Eintrages zur Zeit des Crash, die Prozessorregister und den physikalischen Speicher in eine kontinuierliche Datei, genannt Systemdumpdatei SYSDUMP.DMP der Directory [SYSEXE]. Die Größe der Datei in Blöcken wird mit der Utility SYSGEN generiert und entspricht der Hauptspeichergröße +4 (in Pages). Beim Einschreiben wird der vorhergehende Inhalt der Datei zerstört. Um den alten Inhalt vor dem Überschreiben zu retten, sollte nach Crash eine Kopie unter einem anderen Namen angefertigt werden.

```
$ COPY SYSDUMP.DMP SAVEDUMP.DMP
```

Entsprechend der Dateispezifikation kann diese oder jene Datei analysiert werden.

```
$ ANALYZE/CRASH SYSDUMP.DMP
SDA> SH CRASH
```

Mit SH CRASH werden angezeigt:
- SVP 1800- und Prozeßinformationen wie z.B.
  Datum und Uhrzeit des Crash, Name des Prozesses und des Image, das gerade zur Zeit des Crash bearbeitet wurde, Interrupt-Prioritätsniveau des Prozessors
- Register R0 - R11, AP, FP, SP, PC, PSL
- Prozeß- und Hardware-Maintenance-Register wie Perprozeß- und Systemregister, Stackpointer, Hardware-Maintenance Register

Mit dem Kommando:

```
SDA> HELP      bzw.
SDA> H <Kommandoname>
```

können weitere Kommandos bzw. deren Erläuterungen abgefragt werden.
Alle Kommandos beziehen sich auf die Analyse der Crash-Datei, in die zur Zeit des Crash die Informationen abgespeichert wurden. Nähere Ausführungen dazu sind aus der Dokumentation SVP 1800 zu entnehmen.

1.6.3.3. ANALYZE/SYSTEM

ANALYZE/SYSTEM dient der Analyse des laufenden Systems. Es sind die gleichen Kommandos wie für ANALYZE/CRASH vorhanden, nur daß sie sich nicht auf die Crashdump-Datei, sondern auf das laufende System beziehen. Der Aufruf des Systemdump-Analyzers erfolgt durch

```
$ ANALYZE/SYSTEM
```

Mit

```
SDA> HELP                  bzw.
SDA> HELP <Kommandoname>
```

können weitere Kommandos bzw. deren Erläuterungen abgefragt werden. Nähere Ausführungen dazu sind aus der Dokumentation SVP 1800 (Teil SDA) zu entnehmen.

1.6.3.4. ANALYZE/ERROR_LOG

Das Betriebssystem SVP 1800 schreibt automatisch Fehlernachrichten in die letzte Version der Datei ERRLOG.SYS der Directory [SYS0.SYSERR]. Mittels Kommando

```
$ ANALYZE/ERROR_LOG
```

kann man die Informationen dieser Datei anzeigen.

Folgende Fehler werden in die Datei eingetragen:
- Gerätefehler
- Asynchrone Schreibfehler
- korrigierte Lesedaten
- Hardwarefehler wie z.B. Prioritätsfehler im Cache-Speicher oder in einem Übersetzungspuffer
- Abbruch-Softwarefehler

Darüber hinaus werden Systemereignisse in die Datei eingetragen:
- Normaler Systemstart (Kaltstart)
- Restart nach Spannungsausfall (Warmstart)
- Kaltstart nach Crash (Crashrestart)
- MOUNT- und DISMOUNT-Operationen
- ERROR-LOG-Meldungen, die vom Operator oder vom Prozeß gesendet werden
- Time stamps, die anzeigen, daß sich keine Fehler in einer gegebenen Zeitperiode ereignet haben.

Abhängig davon, wie der Nutzer ANALYZE/ERROR_LOG aufruft, d.h. mit welchen zusätzlichen Schaltern, werden entweder alle Fehler oder Ereignisse der Datei oder eine durch den zusätzlichen Schalter spezifizierte Untermenge von Fehlern und Ereignissen ausgegeben. Mit

```
$ HELP ANALYZE/ERROR_LOG
```

kann man sich über die zusätzlichen Schalter informieren.

Beispiele:

```
$ ANALYZE/ERROR_LOG/SINCE=15-Jul-1987
```
spezifiziert nur jene Einträge der ERROR_LOG-Datei, die später

als die eingegebene Zeit liegen.

```
$ ANALYZE/ERROR_LOG/INCLUDE=(Gerät oder ENTRY-Typ)
Das Gerät kann sein:
- BUSES
- DISKS (z.B. DB als Gerätegruppe, DBA1 Gerät 1)
- REALTIME
- SYNC_COMMUNICATIONS
- TAPES
ENTRY-Typen können sein:
- BUG CHECKS
- DEVICE_ERRORS
- MEMORY
- TIMEOUTS
- MACHINE_CHECK

$ ANALYZE/ERROR_LOG/INCLUDE=MEMORY DRA0:[SYS0.SYSERR]ERRLOG.SYS
$ ANALYZE/ERROR_LOG/INCLUDE=DISK DRA0:[SYS.SYSERR]ERRLOG.SYS
$ ANALYZE/ERROR_LOG/INCLUDE=DRB1/REGISTER_DUMP DRA0:[SYS0.SYSERR]ERRLOG.SYS
```

Die Datei ERRLOG.SYS wächst durch neue Einträge in ihrer Größe, bis sie umbenannt oder gelöscht wird. Um nicht zuviel Platz für die Datei ERRLOG.SYS auf dem Systemplattenspeicher zu binden, könnte z.B. täglich ERRLOG.SYS umbenannt werden.

```
$ RENAME ERRLOG.SYS ERRLOG.DLD
```

Die Datei ERRLOG.DLD könnte dann auf einen Backup-Speicher kopiert werden.

2. Konsoltests

Die Konsoltests ermöglichen eine Fehlersuchstrategie , nach der Fehler des Konsolrechners eingegrenzt werden können. Mit Hilfe der Prüfsystemunterlagen (PSU) kann die Funktionsfähigkeit der Baugruppen nachgewiesen werden. Eine Funktionsprüfung erfolgt durch zyklische Abarbeitung der Prüfprogramme unter Steuerung des Leitprogramms. Die Programme müssen fehlerfrei laufen.
Im Fehlerfall erfolgen Ausgaben, die die Abgrenzung der defekten Baugruppe ermöglichen und genauere Fehlerinformationen geben.
Die Sicherheitsvorschriften des Konsolrechners sind zu berücksichtigen.
Für die Fehlerortung werden als Unterlagen und Hilfsmittel benötigt:

- Prüfprogramme in ladefähiger Form, d. h. auf Diskette
- Servicebaugruppen für die Prüfprogramme
- Vielfachmeßinstrument

2.1. Fehlerortung

2.1.1. Diagnosemöglichkeiten

Im Fehlerfall sind im wesentlichen die folgenden Diagnosemöglichkeiten gegeben:

- Prüfsystemunterlagen (PSU)
- Bedienung mit Bedieneinheit (BDE)

Mit Hilfe der Bedienung mit Bedieneinheit kann der Konsolrechner in seinen einfachsten Funktionen auf einfache Weise überprüft werden.
Für die Diagnose der Baugruppen des Konsolrechners existieren Prüfprogramme, die unter Steuerung des Leitprogramms LPROEM oder autonom abgearbeitet werden können. Sie ermöglichen eine schnelle Überprüfung des Konsolrechners. Auftretende Fehler führen unter Steuerung des Leitprogramms zur Ausgabe von Fehlermeldungen. Bei autonomer Betriebsweise werden Fehlerinformationen aufbereitet und in einem Fehlerfeld bereitgestellt. Die Bedienung des Leitprogramms, die der Prüfprogramme, sowie Angaben zur Bedeutung von Fehlermeldungen und -informationen sind den Abschnitten ab Pkt. 2.2. dieses Kapitels zu entnehmen. Eine Zusammenstellung der Prüfprogramme, die beim Konsolrechner zum Einsatz kommen können, ist in Tabelle 2-1 enthalten.

2.1.2. Fehlersuchstrategie

Bei auftretenden Fehlern wird zunächst überprüft, ob es sich um einmalige Fehler, sogenannte Aussetzfehler, handelte. Fehlerbilder zu Aussetzfehlern werden registriert, um nach mehrmaligem Auftreten eines Fehlers eine zielgerichtete Fehlersuche zu erleichtern.
Liegt kein Hinweis auf Fehlverhalten einer bestimmten Baugruppe vor, so wird zuerst die Funktion der ZVE überprüft. Führt diese Maßnahme nicht zur Fehlermeldung, werden weitere Programme einbezogen.
Wenn der auftretende Fehler auf das Fehlverhalten einer bestimmten Baugruppe hinweist, wird die entsprechende Baugruppe zunächst mit dem zugehörigen

Prüfprogramm kontrolliert.
Fehler, die mittels Prüfprogrammen nicht festgestellt werden, müssen im speziellen Programm weiter verfolgt werden, in dem sie aufgetreten sind.

Tabelle 2-1: Prüfprogramme für den Konsolrechner

lfd. Nr.	Benennung	Kurzzeichen
1.	Leitprogramm	LPROEM
2.	Test der Verzweigungsbefehle	ZVZW20
3.	Test der Arithmetikbefehle	ZARI20
4.	Test der Bedingungsbits	ZBBI20
5.	Test der Verschiebebefehle	ZSHI20
6.	Befehlstest	ZBEF30
7.	Test der Adressierungsarten und Register	ZADRAR
8.	TRAP-und Interrupttest	ZTRAPI
9.	Test der Bussteuerung und ROM-Test	ZBUSCI
10.	Speichertest	ZSPE20
11.	Test der Anschlußsteuerung für serielle Interfaces mit Kurzschlußeinrichtung	AKSAIS
12.	Test der Anschlußsteuerung für Folienspeicher und Laufwerktest	GFLOPM

Bemerkung: Die bezüglich des Konsolrechners angegebenen Adressen sind Oktalzahlen.

2.1.2.1. Fehlersuche ZVE K 2662

Die ZVE besteht aus den Steckeinheiten 051-1130, 051-1140 und 051-1150.
Die Überprüfung der ZVE geschieht durch zyklische Abarbeitung der ZVE-Prüfprogramme (lfd. Nr. 2 9 in Tabelle 2-1) unter Steuerung des Leitprogramms.
Wird von einem Prüfprogramm eine Fehlerausschrift erzeugt oder ist die zyklische Abarbeitung unter Steuerung des Leitprogramms nicht möglich, wird die Anwendung der Prüfprogramme unter autonomer Steuerung versucht.
Versagt auch diese Möglichkeit, kann die weitere Fehlereingrenzung mittels Bediengerät (Punkt 2.1.2.6.) versucht werden.
Die Prüfprogramme der ZVE bauen in ihrer Reihenfolge aufeinander auf. Es wird daher die in Tabelle 2-2 angegebene Reihenfolge empfohlen.
Bei auftretenden Fehlern werden die STE in der angegebenen Reihenfolge getauscht (Tabelle 2-2). Nach jedem Tausch einer STE wird geprüft, ob der Fehler noch auftritt. Tritt der Fehler noch auf, wird die eben getauschte STE wieder eingesetzt

und die in Tabelle 2-2 als nächste angegebene STE ausgetauscht.

Tabelle 2-2: Tauschreihenfolge der STE bei auftretendem Fehler

Programm	STE-Typ
ZVZW20	051-1140; 051-1150
ZARI20	051-1150; 051-1140
ZBBI20	051-1150; 051-1140
ZSHI20	051-1150; 051-1140
ZADRAR	051-1140; 051-1150
ZBEF30	051-1140; 051-1150
ZTRAPI	051-1130; 051-1140; 051-1150
ZBUSCI	051-1130; 051-1140; 051-1150

2.1.2.2. Fehlersuche Speicher OPS K 3564.01

Der Operativspeicher ist eine Steckeinheit vom Typ 051-1730.
Mit Hilfe des Prüfprogramms ZSPE20 kann die Funktionsfähigkeit des Operativspeichers überprüft werden; und zwar im Dauertest der beschreibbare Speicher oberhalb Leitprogramm und Prüfprogramm und im Einzeltest ohne Leitprogramm der ganze Speicher mit Ausnahme des Speicherbereiches des Testprogramms selbst und dessen Arbeits- und Generierbereichs. Die Größe des Speicherbereichs des Prüfprogramms sowie Arbeits- und Generierbereich sind der Beschreibung des Programms (Pkt. 2.9.) zu entnehmen.
In der Fehlerinformation (siehe Prüfprogrammbeschreibung) wird die fehlerhafte Adresse angegeben.
Bei erfolgreichem Dauertest ist der untere Speicherteil indirekt geprüft, da das Leitprogramm jene Adressen für seine Arbeitsfähigkeit benötigt. Ist der Dauertest nicht möglich, kann der angesprochene Einzeltest durchgeführt werden (Pkt. 2.14.3.). Werden Speicherfehler im Testprogramm-, Arbeits- und Generierbereich vermutet, so ist die Speicherkarte gegen eine andere Speicherkarte zu tauschen.
Ist der Fehler nicht auf fehlerhafte Adressen eingrenzbar, so können folgende Fehler vorliegen:

- defekte Spannungsversorgung
- fehlerhafte Ansteuerbedingungen über den Systembus
- fehlerhafte Ansteuerbedingungen für die Speicher-STE

Durch Steckeinheitentausch oder Abarbeitung anderer Prüfprogramme ist der Fehler zu lokalisieren.

2.1.2.3. Fehlersuche für AIS K 8060.02

Die Fehlersuche der Anschlußsteuerung für serielle Interfaces (AIS) ist für den Konsolrechner die Prüfung der STE 051-1760 mit 2mal IFSS- und 1mal S2-Anschluß. Die Testung wird programmtechnisch mittels PSU durchgeführt unter Zuhilfenahme von Kurzschlußeinrichtungen an den griffseitigen Steckverbindern. Für diesen

Zweck ist die AIS (STE 051-1760) des Konsolrechners wie folgt zu konfigurieren:

X3 mit Kurzschlußstecker KIS1
X4 mit Bedieneinheit BDE
X6 mit Kurzschlußleiste KIS3

Der Kurzschlußstecker KIS1 ist eine Leiterplatte vom Typ 051-2000 (Abmessung 50 x 50), die Sender und Empfänger über Lastwiderstände und Leuchtdioden in eine Schleife schaltet.
Die Kurzschlußleiste KIS3, STE 51-2290, ist als 26-polige Steckerleiste realisiert, die die Ausgangssignale an die Eingänge des gleichen Kanals zurückführt.
Über X4 wird der funktionsfähige Rechner mit Bedienmöglichkeit (BDE) realisiert.
Mit dieser Konfiguration wird die AIS bezüglich S2- und eines IFSS-Anschlusses direkt und des weiteren IFSS-Anschlusses indirekt über die Funktion der BDE geprüft. Zur Prüfung wird das Prüfprogramm AKSAIS verwendet (Pkt. 2.12.). Die zu verwendende Generierung ist in Pkt. 2.14.3. notiert. Die Initialisierung der Steckeinheit erfolgt entsprechend den Forderungen der PSU (siehe Pkt. 2.12.) in Übereinstimmung mit den auf der STE festgelegten Parametern.
Fehlerarten:
Es wird unterschieden in Daten-, Interrupt- und Zeitfehler. Spezielle Hinweise zu den Fehlerursachen werden über Bildschirm der BDE ausgegeben. Sie enthalten Angaben zur Kanaladresse und zum fehlerhaften LSI-Schaltkreis. Die Fehlerortung ist damit bis zum geprüften Kanal der Steckeinheit möglich.

2.1.2.4. Fehlersuche für AFS K 5163

Die Anschlußsteuerung für die Folienspeichereinheit (AFS) besteht aus den Steckeinheiten 051-2640, 051-2650 und 051-2660. Die Fehlersuche wird ausschließlich programmtechnisch mittels PSU, Prüfprogramm GFLOPM, durchgeführt. Das Ziel besteht darin, entweder die untersuchte AFS als Fehlerquelle auszuschließen, oder mit auf der BDE angezeigten Fehlerhinweisen über den Wechsel der Steckeinheit zu entscheiden.
Voraussetzung zur sinnvollen Anwendung der PSU ist eine physisch fehlerfreie, formatierte Arbeitsdiskette. Nachdem GFLOPM geladen ist, wird die Arbeitsdiskette ins Laufwerk gegeben.
Da GFLOPM neben der Prüfung der AFS auch die Laufwerkprüfung durchführen kann, ist das Routinesteuerwort entsprechend zu generieren (Pkt. 2.13.). Im Fehlerfalle werden Fehlertexte und -code ausgegeben, deren Interpretation im Pkt. 2.13. erläutert ist.

2.1.2.5. Fehlersuche für BAA

Im Konsolrechner ist als Busabschlußadapter (BAA) die Steckeinheit 057-8070 verwendet.
Mit dem Prüfprogramm ZBUSCI wird der BAA bezüglich der Busleitung geprüft, speziell die Adaptierung zwischen K 1600-Systembus und KBUS.
Die auf dieser Steckeinheit untergebrachte Lüfterausfall-Erkennung wird nach Prüfvorschrift Nr. 1.57.557200.3/12 Teil 1 des BAA geprüft.
Bei Lüfterausfall im Konsolrechner wird dieser stromlos gemacht.

2.1.2.6. Fehlersuche mittels Bediengerät

Die Fehlersuche wird mit dem Bediengerät (BDE) fortgesetzt, wenn diese Bedienung noch funktioniert, die Anwendung der Prüfsystemunterlagen aber nicht möglich ist.

Die Bedienung prüft man wie folgt:

- Einrasten der Taste "HALT/BEREIT" am Frontpaneel,
  Einschalten der Netzspannung —> 173000 erscheint auf der BDE
- Einschalten Netzspannung,
  Einrasten der Taste "HALT/BEREIT" —> Anzeige "HALT" leuchtet
  Drücken der RESET-Taste —> Anzeige "BEREIT" leuchtet kurz auf
  173000 erscheint auf der BDE
- Bedienanweisung
  Der Konsoltechner muß sich dazu im Bedienzustand befinden; dieser Zustand wird dem Bediener durch die Ausgabe des Befehlszählerstandes und des Zeichens "@" über die Bedieneinheit kenntlich gemacht. Der Bedienzustand wird durch Einrasten der Taste "HALT/BEREIT" (Anzeige "HALT" leuchtet) oder durch Betätigen der Taste "BREAK" an der Bedieneinheit erreicht.
  Alle Zifferneingaben und -ausgaben werden oktal dargestellt.
  Als Bedienzeichen werden genutzt: <CR>, <LF>, Leerzeichen (SP), /, 0 7, G, I, P, R, S,], (@, ? nur als Rechnerausgabe).
- Adressierung
  Die Adressierung ist abhängig vom 1. eingegebenen Zeichen, das auf das Anforderungszeichen "@" folgt. Das erste Zeichen ist
  ein "R" (Registerkennzeichen)
  R0 ... R7 wird adressiert, falls eine Ziffer 0 ... 7 nachfolgt; PSW (Prozessorstatuswort) wird adressiert, falls "S" folgt.
  eine Oktalziffer (0 ... 7)
  Die eingegebenen Ziffern werden zu einer 6-stelligen Adresse zusammengesetzt. Bei Eingabe von mehr als 6 Ziffern werden nur die letzten 6 berücksichtigt. Bei Eingabe von weniger als 6 Ziffern werden führende Nullen für eine 6-stellige Adresse ergänzt.
  ein Kommando "G" oder "P"
  Der Inhalt des Befehlszählerstandes (R7) wird als Adresse verwendet und die Programmabarbeitung wird gestartet (G) oder es wird in der Programmabarbeitung fortgefahren (P). Ist die "HALT"-Taste gedrückt, so erfolgt nur die Ausgabe des Befehlszählerinhalts auf der BDE.
- Bedienkommandos
  "/" Laden und Ausgeben
  Die nach "@" eingegebene Adresse (Speicherplatz, Register oder PSW) wird gelesen und der Inhalt angezeigt.
  z. B. @ 007404/021525
  . Eingabe auf Speicherplatz, Register oder PSW
  Nach dem Laden und Lesen einer Adresse wird ein oktales Datenwort als neuer Inhalt der Adresse eingegeben.
  z.B. @ 007404/021525 012242
  <CR> Kommandoabschluß —> Anforderungszeichen
  z.B. @ 006332/012412 1000<CR>
  @
  d.h. der neue Wert ist eingeschrieben,
  Kontrolle durch erneutes Lesen: @ 006332/001000
  oder @ 007200/034112 <CR>
  @
  d.h. der alte Wert bleibt erhalten,
  Kontrolle: @ 007200/034112
  <LF> Kommandoabschluß, Erhöhen der geladenen Adresse um 2 und Anzeige des Inhaltes
  z.B. @ 100/000340 <LF>
  000102/064500
  "]" Kommandoabschluß, Erniedrigen der geladenen Adresse um 2 und Anzeige des Inhaltes
  z.B. @ 334/000240]

```
            000332/063200
"P"  Wiederstart mit Befehlszählerstand (R7)
     Ist die "HALT"-Taste gesetzt, dann wird nur ein Befehlsschritt ausge-
     führt, der Befehlszählerstand ausgegeben und das Anforderungszeichen
     erscheint.
     z.B. @ P
          132402
          @
     Ist die "HALT"-Taste nicht gesetzt, so wird das geladene Programm ab
     dem Befehlszählerstand abgearbeitet.
"G"  Programmstart von der angegebenen Adresse
     z.B. @ 200G
     Wurde keine Adresse eingegeben ,dann Programmstart mit Befehlszähler-
     stand.
     Durch den Start wird das Signal GENLO auf den Bus ausgegeben.
     Ist die "HALT"-Taste gesetzt, erfolgt die Ausgabe des Befehlszähler-
     inhaltes und des Anforderungszeichens.
     z.B. @ 200G
          200
          @
     Ist die "HALT"-Taste nicht gesetzt, so wird das Programm ab der Starta-
     dresse abgearbeitet.
Zur Überprüfung folgender Test:
     Beschreiben der Zellen wie folgt:
          @ 2222/...... 240<LF>
            2224/...... 240<LF>
            2226/...... 137<LF>
            2230/...... 4442<CR>
          @ 4442/...... 0<LF>
            4444/...... 240<LF>
            4446/...... 137<LF>
            4450/...... 6664<CR>
          @ 6664/...... 0<CR>
     Programmstart mit G in 2222 —> Programmhalt in 4444
     Programmfortsetzung mit P —> Programmhalt in 6666
     also:@  2222G
          4444
          @ P
          6666
          @
"I"  Lade Adresse ohne Ausgabe
     Dieses Kommando wird zum Beschreiben nicht lesbarer Zellen verwendet.
     z.B. @ 177170/?
     d.h. diese Geräteadresse ist vor der Initialisierung nicht lesbar.
     Anwendung zur Initialisierung einer im Konsolrechner vorhandenen
     Anschlußsteuerung
     z.B. Initialisierung der AFS (Zelle 177170 nicht lesbar)
     @177760 I 173574 <CR>
     @177762 I 177172 <CR>
     @177764 I  70264 <CR>
     nun muß Zelle 177170 lesbar sein
```

So können vom Bediengerät aus am Systembus angeschlossene Komponenten des Rechners und die internen Register der ZVE in ihren Grundfunktionen überprüft werden. Gleichzeitig werden elementare Funktionen der ZVE-Steckeinheiten 051-1140, 051-1150 und des Konsolrechner-Frontpaneels (FP-LP) BLP 051-1664 kontrolliert. Dazu werden die Speicherzellen und internen Register mit verschiedenen Testmustern beschrieben und anschließend gelesen.

Die Eingabe von einfachen Befehlsfolgen, die für den jeweiligen Fehlerfall als kritisch angesehen werden, ist möglich. Ihre Abarbeitung kann in Verbindung mit der Taste "HALT/BEREIT" am Frontpaneel befehlsweise erfolgen.
Auch das Ladeprogramm wird befehlsweise abgearbeitet, wenn die Anwendung der PSU daran scheitert. Mit Hilfe der Befehlsfolgeliste des Ladeprogramms ist die Fehlereingrenzung möglich.

Befehlsfolgeliste des Anfangsladers fuer Handeingabe

- Voraussetzung: Initialisieren der AFS
- Eingabe nachstehender Befehlsfolge ab Adresse 1000 im Bedienzustand

```
@1000/......   5000<LF>
 1002/......  12704<LF>
 1004/...... 177170<LF>
 1006/......  12705<LF>
 1010/...... 177172<LF>
 1012/......  12714<LF>
 1014/......  40000<LF>
 1016/......  32714<LF>
 1020/......     40<LF>
 1022/......   1775<LF>
 1024/......  12714<LF>
 1026/......   20X7<LF>   X=0 : LW0 ,  X=2 : LW1
 1030/......  12703<LF>
 1032/......      2<LF>
 1034/......  32714<LF>
 1036/...... 100200<LF>
 1040/......   1775<LF>
 1042/...... 100435<LF>
 1044/......  12715<LF>
 1046/......      1<LF>
 1050/......   5303<LF>
 1052/......   1370<LF>
 1054/......  32714<LF>
 1056/...... 100040<LF>
 1060/......   1775<LF>
 1062/...... 100425<LF>
 1064/...... 112737<LF>
 1066/......     X0<LF>   X wie oben
 1070/......  47772<LF>
 1072/......  12714<LF>
 1074/......      3<LF>
 1076/......  32714<LF>
 1100/...... 100200<LF>
 1102/......   1775<LF>
 1104/...... 100414<LF>
 1106/......  12715<LF>
 1110/......    100<LF>
 1112/......  32714<LF>
 1114/...... 100200<LF>
 1116/......   1775<LF>
 1120/...... 100406<LF>
 1122/......  10015<LF>
 1124/......  32714<LF>
 1126/...... 100040<LF>
 1130/......   1775<LF>
 1132/...... 100401<LF>
```

```
1134/......    401<LF>
1136/......      0<LF>   (Fehlerhalt)
1140/......   5000<LF>
1142/......   5007<LF>
1144/......      0<LF>
1146/......      0<LF>
```

- Start des Anfangsladers mit @1000G

2.1.2.7. Prüfung der Betriebsspannungen und der elektrischen Verbindungen

Diese Prüfung wird zweckmäßig vorgenommen, wenn weder die Abarbeitung der Prüfprogramme noch die Bedienung des Rechners möglich ist. Weiterhin werden in diesem Falle die elektrischen Verbindungen überprüft:

- Sitz der STE
- Sitz der griffseitigen Adapter zwischen den einzelnen STE
- Anschlußkabel auf der Griffseite der STE

2.2. Beschreibung des Leitprogramms LPROEM

2.2.1. Benennung

Leitprogramm für Prüfprogramme und Gerätetestprogramme des Konsolrechners.

Bem: LPROEM ist eine modifizierte Variante des LPROES des K 1600-Systems. Es wird in der Beschreibung daher auch nur auf die für den Konsolrechner relevanten Sachverhalte und Gerätekonfigurationen eingegangen.

2.2.2. Dokumentationsnummer

Die Dokumentationsnummer für das auf der Diskette DS0 geladene Leitprogramm LPROEM lautet 1.57.700200.3 , die dazugehörige Programmliste hat die Dokumentationsnummer 1.57.700225.3 .

2.2.3. Programmumfang

LPRVOR	-	Konfigurationsbestimmung, Bedienereingabe, Grundeinstellung
LINTBH	-	Verbindungszellen für Unterbrechungen laden, allgemeine Interruptbehandlung
LPDAUT	-	Dauertestgenerierung, Dauertestabarbeitung
LTPRUF	-	Testprogrammaufrufvorbereitung
LPFEBH	-	Fehlerbehandlung, Fehlerausgabe
LPDIEN	-	Generierungsein/ausgabe über externes Gerät, Generierungsanzeige und Korrektur. Einlesen von Prüfprogrammen, Speicherbereichsausgabe
LBDFSE	-	Bedienroutinen und Dateiorganisation für Folienspeicher

2.2.4. Speicherbedarf

	2400	Stackbereich für LPROEM und Prüfprogramm
2400 ...	4776	Arbeitsbereich LPROEM
5000 ...	6776	Arbeitsbereich für Prüfprogramme
7000 ...	7776	Verbindungsbereich LPROEM-Prüfprogramme
10000 ...	13776	Dauertestgenerierbereich
14000 ...	17776	Datentransferbereich für Prüfprogramme
20000 ...	65400	Programmbereich LPROEM
65400 ...	137776	freier Bereich für Prüfprogramme

2.2.5. Ausführungszeit

entfällt

2.2.6. Testhilfsmittel/Geräteeinstellungen

Das LPROEM benötigt als Peripherie unbedingt ein Bediengerät. Das ist beim Konsolrechner die Bildschirmeinheit (E/A-Adresse:177500).
Als externes Speichergerät ist eine Folienspeichereinheit (E/A-Adresse:177170) vorgesehen.

2.2.7. Bedienungsanleitung für LPROEM

2.2.7.1. Aufruf des LPROEM

Der Aufruf des LPROEM erfolgt automatisch nach Laden des Programmes vom externen Speichergerät. Als Standardbediengerät ist die Bildschirmeinheit K 8911 vereinbart. Weitere Startmöglichkeiten aus dem Bedienzustand ergeben sich über nachfolgende Start- und Haltadressen:

Adresse	Symbol	Bedeutung
20000	LK16BS	Neustart für Bedienung über K 8911
20026	LKWIST	Wiederstart LPROEM nach Unterbrechung durch Bedienzustand bei Erhaltung vorhandener Generierungen
20034	LPEHLT	angezeigte Haltadresse für generierten Fehlerhalt; Fortsetzung mit "P"
20042	LNETZF	angezeigte Haltadresse bei Netzausfall, wenn kein Restart vorgesehen ist.

Das LPROEM meldet sich mit der Ausschrift LPROEM VERS. "n". Angabe des ZE-Typs und der Ausgabe des verfügbaren Speicherbereichs.
Die weitere Ablaufsteuerung erfolgt durch Bedienereingaben.

2.2.7.2. Steuerkennzeichen für Bedienereingaben

Zur Kennzeichnung der jeweiligen erwarteten Bedienereingabe werden vom LPROEM folgende Steuerkennzeichen ausgegeben:

> erfordert Eingabe eines entsprechend KROS 4034 vereinbarten Prüfprogrammnamens oder eines LPROEM-eigenen Steuerroutinenamens, bestehend aus 6 ASCII-Zeichen zum Aufruf desselben (s. 2.2.7.3.)

>> erfordert Eingabe eines Prüfprogrammnamens zur Dauertestgenerierung

erfordert Eingabe eines aus zwei ASCII-Zeichen bestehenden Steuerkommandos (s. 2.2.7.4.). <CR> beendet die Steuerkommandoeingabe

= erfordert die Eingabe einer oder mehrerer 0 - 5stelliger Oktalzahlen entsprechend der Bedeutung des vorangegangenen Steuerkennzeichens, jede Oktalzahl ist durch Komma bzw. die letzte eines Oktalzahlblocks durch E abzuschließen. Bei Oktaleingaben erfolgt eine Gültigkeitsprüfung. Bei Ungültigkeit oder Überlauf wird das Fehlerkennzeichen F= ausgegeben. Danach kann die Eingabe der Oktalzahl wiederholt werden.

(erfordert die Eingabe von 0 bis 64 Oktalworten zur Dauertestgenerierung. Die Oktalworte sind durch Komma zu trennen. Das letzte Oktalwort ist mit) abzuschließen und beendet die Dauertestgenerierung für das entsprechende Prüfprogramm. Für das letzte zu generierende Prüfprogramm ist die letzte Oktaleingabe mit <CR> anstelle) abzuschließen (s. 2.2.8.4.1.).

erfordert die Eingabe eines entsprechend der vorstehenden Ausschrift vereinbarten Zeichens.

2.2.7.3. Aufruf LPRO-eigene Steuerroutinen

LPRO-eigene Steuerroutinen werden wie Prüfprogramme durch ihren aus 6 ASCII-Zeichen bestehenden Namen aufgerufen und erfüllen folgende Funktionen:

LTKONF Ermittlung der Speicherkonfiguration des Rechners sowie der geladenen Prüfprogrammkonfiguration und Erstellung der Tabelle der Ansprungadressen der Prüfprogramme einschließlich des angewählten externen Speichers.

DAUGEN Dauertestgenerierung. Hiermit wird die durchzuführende Dauertestreihenfolge der Prüfprogramme sowie die spezifische Generierung derselben ermöglicht.

DAUTST Aufruf der generierten Dauertestreihenfolge. Die Protokollsteuerung erfolgt über die unter 2.2.7.4.1. aufgeführten Steuerkommandos.

DAUWEI Fortführung Dauertest nach Testabbruch oder Bedienerhalt, wenn LPROEM über LKWIST gestartet worden ist. Der Dauertest wird nach dem unterbrochenen Prüfprogramm fortgesetzt. Vorher ist eine Veränderung der Protokollsteuerung entsprechend 2.2.7.4.1. möglich.

DSPAUS Speicherausgabe über generiertes Fehlerprotokollgerät. Durch Eingabe von Anfangsadresse nach A= und Endadresse nach E= können beliebige Speicherinhalte ausgegeben werden.

DGNANZ Anzeige- und Korrekturmöglichkeit für eine vorhandene Dauertestgenerierung entsprechend den unter 2.2.7.4.3. beschriebenen Steuerkommandos. Bei Programmen, für die der Standardfall generiert wurde, ist eine Korrektur bzw. Änderung der Generierung nicht möglich.

DGNAUS Ausgabe der Dauertestgenerierung auf das konfigurierte externe Speichergerät bzw. Lochstreifenstanzer. Es sind maximal 16 verschiedene Generierungen auf dem externen Speichergerät unter den Oktalnummern 1 ... 20 abspeicherbar.

DGNEIN Eingabe der Dauertestgenerierung vom externen Speichergerät

LDOPPR Doppeln Datenträger. Das angewählte Speichergerät wird als Datenqelle, das jeweilige verbleibende Gerät der angewählten Speichereinheit als Datensenke benutzt.

LAFZLP Aufzeichnen LPROEM. Das im Hauptspeicher befindliche Leitprogramm wird auf den angewählten Datenträger auf fest zugeordnete Plätze aufgezeichnet.

2.2.7.4. Steuerkommandos

2.2.7.4.1. Steuerkommandos zur Festlegung der Gerätekonfiguration

Bei Kommandos, die sich gegenseitig in der Wirkung ausschließen, ist das zuletzt eingegebene Kommando das für den nachfolgenden Ablauf wirksame.

FS Auswahl des Folienspeichers als externes Speichergerät. Nachfolgend ist entsprechend Aufforderung die Laufwerkadresse (0;1) einzugeben.

BS Bildschirm als Protokollgerät - Rückschaltung von Drucker auf Bedienterminal

Der Konsolrechner erlaubt nur die Arbeit mit dem Folienspeicher als externes Speichergerät.

Die Beendigung der Gerätekonfigurationsfestlegung durch eine Auswahl vorstehender Kommandos erfolgt durch Eingabe von <CR>.

2.2.7.4.2. Steuerkommandos zur Generierung des Einzeltests von Prüfprogrammen

Die nachfolgend aufgeführten Steuerkommandos erfordern zusätzlich die Eingabe einer oder mehrerer Oktalzahlen, die die entsprechende Generierinformation enthalten.

PS Programmstatuswort PSW

RS Routinesteuerwort, Auswahl der jeweils durchzuführenden Routinen

PZ Programmzykluszähler

RZ Routinezykluszähler

GN Prüfprogrammspezifische Generierung. Die prüfprogrammspezifische Generierung ist entsprechend der Reihenfolge der im Prüfprogramm enthaltenen Standardgenerierung vorzunehmen.

Diese Steuerkommandos sind nur nach Aufruf eines Prüfprogramms im Einzelfall gültig.

2.2.7.4.3. Steuerkommandos zum Aufruf und zur Protokollsteuerung für Einzel- und Dauertest

PN Fehlerprotokoll - Ausschrift einer Fehlernummer für jeden Einzelfehler

NP kein Fehlerprotokoll - es erfolgt nur eine Aufsummierung der Fehler für jedes Testprogramm.

QT Endquittung END mit Angabe der Fehlersumme und Fehlerprotokoll

ST Start Einzel- oder Dauertest. Schließt die Steuerzeicheneingabe ab und ruft den angewählten Test auf.

SH Start mit Fehlerhalt nach Ausschrift, sonst wirksam wie ST. Durch Eingabe von <CR> anstelle eines Startkommandos wird zur LPROEM-Bedienung zurückgekehrt.

2.2.7.4.4. Steuerkommandos zur Generierungsanzeige und Korrektur

Nachfolgende Kommandos dienen zur Korrektur bzw. Veränderung oder Ausschrift einer vorhandenen Dauertestgenerierung, die über DAUGEN erstellt wurde oder über DGNEIN von einem externen Gerät eingegeben wurde. Sie sind nur nach Aufruf von DGNANZ wirksam.

AN Ausgabe der vorhandenen Dauertestgenerierungen mit laufender Nummer, Programmnamen und Generierfeldinhalt auf das festgelegte Protokollgerät (Standardgenerierungen enthalten im Generierfeld eine Null).

KR Korrektur einer durch die laufende Nummer spezifizierten Dauertestgenerierung. Die Korrektur erfolgt, indem nach der Ausschrift KORREKTUR DAUGEN PLATZNR./INF.= die Platznummer der zu korrigierenden Information,

Schrägstrich und die neue Information durch Komma abgeschlossen, einzugeben ist. Beendet wird die Korrektur durch Eingabe von E anstelle einer Korrektureingabe. Standardgenerierungen sind nicht korrigierbar.

LO Löschen Dauertestgenerierung. Durch Angabe der laufenden Nummer nach LO wird das dieser Nummer zugeordnete Prüfprogramm aus der Abarbeitungsreihenfolge herausgenommen. Bei Anzeige über AN erscheint anstelle des Programmnamens des gelöschten Programms der Name DAUTST. Eine gelöschte Generierung kann nicht wieder eingetragen werden, sondern ist über "ER" am Ende des Generierfeldes neu vorzunehmen.

ER Erweiterung einer vorhandenen Dauertestgenerierung um weitere Prüfprogramme (die Generierung erfolgt wie für DAUGEN beschrieben).

Die Beendigung einer Korrektur erfolgt durch Eingabe von <CR>.

2.2.7.4.5. Steuerkommandos zur Aufzeichnung von Prüfprogrammen auf externe Speicher

Aufgrund der Gerätekonfiguration des Konsolrechners ist hier nur das die Foliespeichereinheit betreffende Kommando angegeben.

AW Auswechseln Prüfprogramm
Das angewählte Prüfprogramm wird auf die aktive Diskette auf die Position gebracht, die für dieses Prüfprogramm durch vorhergehendes Aufzeichnen vereinbart worden war.

2.2.7.5. Durch LPROEM erzeugte Fehlerausschriften

2.2.7.5.1. Durch Bedienfehler erzeugte Ausschriften

PNAME NICHT DEFINIERT	Eingabe eines nicht vereinbarten Programmnamens.
PNAME NICHT GELADEN	Aufgerufenes Prüfprogramm befindet sich nicht im Speicher. Im Dauertest wird das nachfolgende Programm aufgerufen. Im Einzeltest erneute Eingabe eines Programmnamens
XX NICHT DEFINIERT	Steuerkommando nicht vereinbart - neue Steuerkommandoeingabe
GENERIERFEHLER PNAME	Fehler bei der Programmgenerierung Genaue Spezifikation durch nachfolgende Fehlerausschriften:
UEBERSCHREITUNG GENERIERBEREICH PRUEFPROGRAMM	Es sind mehr als im Prüfprogramm zugelassene Generiereingaben erfolgt.
ENDE GENERIER-BEREICH	Der Gesamtgenerierbereich für Dauertest wurde überschritten.
XX NICHT KONFIGURIERT	Das mit XX bezeichnete Gerät ist nicht in der Rechnerkonfiguration enthalten.

2.2.7.5.2. Fehlerausschriften, die durch Fehlfunktionen des Rechners oder der peripheren Geräte erzeugt werden

RESTART LPROEM	Restart nach vorangegangenem Netzausfall beim Konsolrechner mit Datenerhalt.

DRUCKERFEHLER Gerätefehler angewähltes Protokollgerät, Umschaltung der Protokollausgabe auf Bediengerät.

IN PNAME PRXX R0 R1 R2 R3 R4 R5
SZ SV PS SP PC IV T

Ausschrift der Fehlerform 0 entsprechend Pkt.2.2.9.3. für alle durch das LPROEM erkannten Unterbrechungen. Es erfolgt Abbruch des laufenden Prüfprogramms. Für alle nicht in einer speziellen Prüfroutine aufgetretenen Fehler erfolgt für RTXX die Ausschrift RTOO. IV gibt den Unterbrechungsvektor an.

EE PNAME RTXX LUEFTERAUSFALL ZE
Ausschrift durch Unterbrechung eines laufenden Prüfprogramms durch Lüfterausfall des Rechners.

PRUEFSUMMENFEHLER Festgestellter Prüfsummenfehler bei Programmübertragung vom externen Speicher.

"Gerätename" NICHT BEREIT
Angewähltes externes Speichergerät nicht funktionsfähig.

NETZFEHLER FSE Netzausfall der Folienspeichereinheit. Gerät ist über Netzschalter wieder einzuschalten.

ERW. STATUS FSE Erweiterter Status FSE. Auswertung entsprechend Gerätedokumentation. Rückschlüsse auf Zustand der verwendeten Diskette möglich.

LESEFEHLER CRC Nach 10 vergeblichen Leseversuchen wird das entsprechende Programm ungültig gemacht und ist neu aufzuzeichnen.

LESEFEHLER DISKETTE
Prüfsummenfehler. Nach entsprechender Anzahl von Lesewiederholungen wird das Programm auf der Diskette ungültig gemacht und ist neu aufzuzeichnen.

ROUTINESTEUERWORT UNGUELTIG
Es wurden im Prüfprogramm nicht enthaltene Routinen generiert. Bei allen Generierfehlern wird zur Programmnameneingabe des LPROEM zurückgekehrt.

F= Fehler bei Oktalworteingabe. Überlauf oder falsches Zeichen. Nachfolgend Wiederholung der Oktalworteingabe möglich.

UEBERSCHREITUNG PSU-SPEICHERBEREICH
Bei Laden von Prüfprogrammen vom externen Speicher bzw. Lochband wird der in der Konfigurationsfestlegung angegebene Speicherbereich überschritten. Das Laden wird abgebrochen.

DISKETTE ENTHAELT KEINE PSU
Es wurde eine noch nicht bzw. nicht mit Prüfprogrammen belegte Diskette eingelegt.

DISKETTE VOLL Das aufzuzeichnende Prüfprogramm überschreitet die Speicherkapazität der angewählten Diskette.

PNAME AUF DISKETTE NICHT GELADEN

Das aufgerufene Programm ist auf der Diskette nicht vorhanden bzw. nach 10 vorangegangenen erfolglosen Leseversuchen ungültig gemacht worden.

2.2.7.6. Nutzung der Bedieneinheit K 8911

2.2.7.6.1. Bildschirmanzeige

Bei Start des LPROEM wird der Bildschirm gelöscht und ab Zeile 1 in den Positionen 1 - 39 der Zeilen 1 - 10 ein feststehender Text zur Identifikation des LPROEM sowie Informationen zur Speicher-, Geräte- und Testkonfiguration ausgeschrieben.
Auf den Positionen 40 - 80 der Zeilen 1 bis 10 wird auf feststehenden Plätzen der jeweilige Testzustand angezeigt. Ab Zeile 11 bis Bildschirmende wird der Bildschirm in den Rollmodus zur fortlaufenden Testprotokollierung geschaltet. Die Umschaltung in den Normalmodus des Bildschirms am Ende der Arbeit mit PSU erfolgt durch Eingabe von CTRL/C nach >

2.2.7.6.2. Steuerung des Testablaufes durch Funktionstasten der Bedieneinheit K 8911

Zusätzlich zu den unter 2.2.7.4. beschriebenen Steuerkommandos können nachfolgende Funktionstasten der BDE zur Testablaufsteuerung verwendet werden:

Taste	Funktion
HOLD	Ein/Ausschalten HOLD SCREEN MODUS Es wird eine Bildschirmseite vollgeschrieben, danach geht die BDE in den Wartezustand.
NEXT PAGE	Ausgabe der nächsten Bildschirmseite im HOLD SCREEN MODUS.
CTRL/HOLD	Einschalten des langsamen Rollmodus (SMOOTH ROLL MODUS). Ausschaltbar durch Taste HOLD.
CTRL/REC	Einschalten PRINT LINE MODUS bei angeschlossenem Drucker.
CTRL/REC	Ausschalten PRINT LINE
CLEAR	Einstellen des Grundzustandes der BDE mit Kursor am Bildanfang.
RESET	Rücksetzen BDE bei Fehler.

Alle anderen Funktionstasten können entsprechend Beschreibung der BDE K 8911 verwendet werden, da sie vom LPROEM nicht als Zeicheneingabe verwendet wurden. Es ist zu beachten, daß der PRINT LINE MODUS im HALT (Bedienzustand des Rechners) zu Fehlfunktionen des Druckers (fortlaufender Zeilenvorschub) führt.

2.2.8. Prüfprogrammbearbeitung über LPROEM

2.2.8.1. Anschlußbedingungen der Prüfprogramme

Der Anschluß der Prüfprogramme an LPROEM wird durch einen einheitlichen Programmvorspann realisiert, der Bestandteil jedes Prüf- oder Gerätetestprogramms ist. Dadurch wird für alle angeschlossenen Programme einheitlich die

Generierung und zyklische Abarbeitung im Einzel- oder Dauertest sowie der generierbare Aufruf einzelner Prüfroutinen realisiert. Fehlerprotokollierung sowie alle Textausgaben erfolgen über LPROEM entsprechend der generierten Protokollierungsart.

2.2.8.2. Anfangseinstellung des LPROEM

Durch den Aufruf des LPROEM wird folgende Anfangseinstellung vorgenommen:

- Bediengerät BDE K 8911 gleichzeitig Protokollgerät
- externer Programmspeicher in Grundstellung (z. B. Bereitschaft Folienspeichereinheit geprüft).
- Generierung für Prüfprogramme im Standardfall
- PSU-Speicherbereich LPROEM ENDE ... 137776
- keine Endequittung/keine Fehlerprotokollierung

Durch den Neustart von LPROEM wird die Speicher- und Testprogrammkonfiguration einschließlich des angewählten externen Speichergerätes bestimmt. Hierdurch wird auch die Tabelle der Anfangsadressen für die geladenen Programme erstellt. Vorhandene Einzel- und Dauertestgenerierungen im Hauptspeicher werden gelöscht. Durch Aufruf eines Prüfprogrammes wird die Bedienungsanleitung LPROEM überschrieben.

2.2.8.3. Durchführung Einzeltest

Der Einzeltestaufruf eines Prüfprogramms erfolgt durch Eingabe des Prüfprogrammnamens nach >. Das LPROEM ermittelt den Programmstandort des Prüfprogramms nach dem Prüfprogrammkennzeichen.
Ist das aufgerufene Prüfprogramm nicht im Hauptspeicher vorhanden, wird es vom angewählten externen Speicher geladen. Nach dem Aufruf ist über die unter 2.2.7.4. aufgeführten Steuerkennzeichen die Programmgenerierung vorzunehmen. Eine einmal vorgenommene Generierung bleibt erhalten und wird nur durch erneute Generierung überschrieben, so daß nur jeweils die für den nächsten Testlauf zu verändernden Informationen neu zu generieren sind.
Bei Neustart des LPROEM wird die Anfangseinstellung (s.2.2.8.2.) vorgenommen. Überschreitet das zu ladende Programm den vorhandenen PSU-Speicherbereich, wird das Programm nicht geladen, sondern eine Fehlerausschrift entsprechend 2.2.7.5.1. erzeugt, die PSU-Bereichsanfangsadresse eingestellt und zur Bedienung zurückgekehrt.

2.2.8.4. Durchführung Dauertest

2.2.8.4.1. Dauertestgenerierung

Die Prüfprogramme sind für den Dauertest in beliebiger Reihenfolge ein- oder mehrmals mit unterschiedlichen Generierinformationen generierbar.
Insgesamt sind 64 Programme für einen Dauertest generierbar. Der Generierungsaufruf erfolgt über DAUGEN und wird durch <CR> anstelle) nach Generierung des letzten Prüfprogramms abgeschlossen. Eine eingegebene Dauertestgenerierung kann über DGNANZ entsprechend 2.2.7.4.4. korrigiert bzw. erweitert werden. Über DGNAUS kann diese Generierung auf ein externes Speichergerät aufgezeichnet werden. Über DGNEIN kann diese Generierung wieder zur Abarbeitung verfügbar gemacht werden. Insgesamt sind 16 Dauertestgenerierungen, gekennzeichnet durch die bei der Aufzeichnung zugeordnete Oktalnummer (1...20), aufzeichenbar. Die Aufzeichnung von Dauertestgenerierungen erfolgt automatisch auf das angewählte externe

Speichergerät.
Die Dauertestgenerierung ist im einzelnen wie folgt vorzunehmen:
Eingabe des Prüfprogrammnamens nach >> und nach "(" Eingabe der allgemeinen und prüfprogrammspezifischen Generierinformationen in folgender Reihenfolge als Oktalworte:
- aktuelles PSW
- Dauertestroutinesteuerwort
- Zykluszähler Prüfprogramm
- Zykluszähler Routine
- Eingabe der prüfprogrammspezifischen Generierinformationen in der im Prüfprogramm vereinbarten Reihenfolge.

Wenn der im Prüfprogramm definierte Standardfall zur Anwendung kommt, ist sofort nach "(" mit ")" die Eingabe zu beenden.
Sind Generierungen vorzunehmen, werden bis zur letzten, vom Standardfall abweichenden Generierinformation, alle Generierinformationen in der vorgeschriebenen Reihenfolge eingetragen.

2.2.8.4.2. Dauertestabarbeitung

Die Dauertestabarbeitung wird durch DAUTST angewiesen. Danach kann entsprechend 2.2.7.4.1. die Protokollsteuerung angewiesen werden. Nach ST meldet sich das LPROEM mit DAUERTEST START und die Prüfprogramme werden in der generierten Reihenfolge abgearbeitet.
Nach Ende eines Durchlaufes erfolgt die Meldung wieder mit DAUERTEST START und Ausgabe des Durchlaufzählers sowie bei aufgetretenen Fehlern des Fehlerzählers, der die Fehler fortlaufend aufsummiert. Die Dauertestabarbeitung erfolgt endlos. Unterbrechungen können durch Bedienerhalt vorgenommen werden. Durch Wiederstart des LPROEM über Adresse 20026 und Eingabe von DAUWEI nach > wird der Dauertest nach dem unterbrochenen Programm fortgeführt. Eine Veränderung der Protokollsteuerung und Gerätekonfiguration nach DAUWEI ist erlaubt. Die angewählten Programme werden zuerst im Hauptspeicher gesucht und nur wenn dort nicht vorhanden, vom angewählten externen Speichergerät geladen. Ist der zur Verfügung stehende PSU-Speicherbereich für das zu ladende Programm nicht mehr ausreichend, so wird das Programm ab der Anfangsadresse des PSU-Speicherbereiches geladen und die dort befindlichen Programme überschrieben. Ist das zu ladende Programm größer als der zur Verfügung stehende Speicherbereich, wird der Ladevorgang und der Dauertest mit entsprechender Fehlermitteilung abgebrochen.

2.2.8.5. Unterbrechungsbehandlung durch LPROEM

2.2.8.5.1. Allgemeine Unterbrechungsbehandlung

Durch das LPROEM werden alle Unterbrechungsquellen, die nicht vom jeweiligen Prüfprogramm bedient werden, behandelt. Als Maßnahme wird eine Fehlerausschrift mit Ausgabe aller Register und der Unterbrechungsvektoradresse in IV ausgegeben (s. 2.2.7.5.2.). Danach erfolgt der Abbruch des laufenden Prüfprogramms. Im Dauertest wird das nachfolgende Programm aufgerufen. Im Einzeltest kann über die Bedienereingabe ein neuer Programmaufruf erfolgen.

2.2.8.5.2. Netzfehler- und Restartbehandlung

Bei Netzausfall werden alle wichtigen Arbeitsregister auf einen Speicherbereich, der Datenerhalt garantiert, gerettet. Anschließend geht der Rechner in den Halt. Über die Restartroutine nach Wiederstart des Rechners über die Verbindungszelle 24 werden alle geretteten Register rückgespeichert und RESTART LPROEM

ausgegeben. Im Dauertest wird das dem unterbrochenen Programm nachfolgende Programm aufgerufen, da nicht sichergestellt werden kann, daß nach Netzausfall der Gerätezustand mit dem unterbrochenen Zustand identisch ist. In allen anderen LPROEM-Arbeitsweisen wird der Grundzustand hergestellt und eine Bedieneranweisung erwartet.

2.2.9. Fehlerprotokollierung

2.2.9.1. Aufbau der Fehlerausschriften

Alle Fehlerausschriften der Prüfprogramme werden im LPROEM realisiert. Im Prüfprogramm erfolgt die Festlegung der Form der Fehlerausschriften durch eine Trapcodierung entsprechend den Zuordnungen in Tabelle 4 und 5 der Richtlinie KROS 4034. Eine Fehlerausschrift besteht aus den nachfolgenden aufgeführten Teilen.

- Fehlerkennzeichen	2 ASCII-Zeichen zur Kennzeichnung der Fehlerart s. 2.2.9.2.1
- Programmname	6 ASCII-Zeichen, enthält den entsprechend KROS 4034 vereinbarten Prüfprogrammnamen
- Routinenummer	RTXX enthält die oktale Codierung der aktuellen Routinenummer des laufenden Prüfprogramms
- Fehlerinformationen	5 oder 10 Oktalwerte, deren Bedeutung durch zwei voranstehende ASCII-Zeichen (s. 2.2.9.2.) gekennzeichnet ist bzw. Textinformationen max. 38 Zeichen oder eine ASCII-Zeichengruppe als Statusmeldung serieller Geräte, s. 2.2.9.3. Fehlerformen
- Abbruchkennzeichen	T wird am Ende einer Fehlermitteilung ausgegeben, wenn durch diese Abbruch des Prüfprogramms angewiesen wurde.

Zusätzlich zu den Fehlerausschriften kann eine erläuternde Textausgabe oder eine Ausgabe eines Speicherblockes auf den der Fehlerausschrift nachfolgenden Protokollzeilen erfolgen, deren Bedeutung den Prüfprogrammbeschreibungen zu entnehmen ist.

2.2.9.2. Bedeutung der Kurzzeichen in Fehlerausschriften

Durch die als Fehlerkennzeichen und die zur Kennzeichnung von Fehlerinformationen verwendeten Kurzzeichen soll eine grobe Fehlerwertung ohne zusätzliche Hilfsmittel ermöglicht werden.

2.2.9.2.1. Fehlerkennzeichen

AF Adressenfehler, Speicher/Geräteadresse nicht vorhanden oder fehlerhaft

BD Bedienfehler, falsche Gerätebedienung, falsche Programmgenerierung

BF Befehlsablauf fehlerhaft

DP Datenfehler, Ist/Solldatenvergleich

GF Gerätefehler, AS/Gerät nicht funktionsfähig

IF Interruptfehler, Interrupt auf falschem Kanal oder ohne Ursache

MF Maskierungsfehler, Interrupt bei gesperrter Interruptmaske

PA Paritätsfehler

SF Statusfehler, Kurzschlußtest Status Ist/Sollvergleich
nicht erwarteter Fehlerstatus vom Gerät

SP Speicherfehler

SK Speicherfehler bei Speicher mit Fehlerkorrektur
Ausgabe des Fehlerspeicherinhaltes

GK Gleitkommafehler - Arithmetikprozessor

ZF Zeitfehler, Wartezeit auf Unterbrechung, Wartezeit bei Gerätebedienung
im Abfragebetrieb

DB Datenblockfehler. Nach der Fehlerausschrift wird ein Datenblock aus dem
Speicher mit zugehöriger Speicheradresse zur Fehlerermittlung oktal
ausgegeben.

IN Interner Fehler, vom LPROEM als Kennzeichnung für alle Ausschriften,
die durch vom LPROEM erkannte Unterbrechungen erzeugt werden, verwendet.

EE Eingriff erforderlich, Gerätestatusmeldungen, die eine Gerätebedienung
erfordern wie z. B. Bandriß, Papier alle usw.

2.2.9.2.2. Kennzeichen zur Deutung der Fehlerinformationen

AD Adresse, Speicher-, Anschlußsteuereinheit- oder Geräteadresse, die im
ursächlichen Zusammenhang mit dem Fehler steht.

BA Blockanfangsadresse bei Datenblockfehler

BC Befehlscode eines durch BF als fehlerhaft gekennzeichneten Befehls

BE Blockendeadresse bei Datenblockfehler

Cn Statusregister bei verschiedenen Anschlußsteuereinheiten (n = 1 8)

Dn Datenwörter im Fehlerbereich zu übergeben (n = 1 6)

IS Istwert bei Ist/Sollwertvergleich aktueller Testinformationen
(E/A-Daten, Status)

IV Interruptvektoradresse, enthält in den Bits 0 ... 8 die Vektoradresse der
mit dem Fehler in ursächlichem Zusammenhang stehenden Unterbrechung

PC Befehlszähler zum Fehlerzeitpunkt

PS Prozessorstatuswort zum Fehlerzeitpunkt

Rn Allgemeine Register werden dann ausgegeben, wenn keine
Informationszuordnung möglich ist (n = 0 5)

SA Speicheradresse 18 Bit einschließlich der zwei Adreßbits der
Speichererweiterung

SC Statusmeldung serieller Geräte und Anschlußsteuereinheiten in ASCII-Zeichen 1 ... 9 Zeichen nachfolgend möglich

SL Sollwert bei Ist/Sollwertvergleich aktueller Testinformationen

SP Stapelzeiger (Stackpointer) zum Zeitpunkt des Fehlers

SZ ZE-Status

T Testabbruchkennzeichen, wird nach einer Fehlerausschrift ausgegeben, wenn anschließend das laufende Prüfprogramm abgebrochen wird. Bei programmiertem Fehlerhalt erfolgt die Ausschrift "T" erst nach nachfolgender Programmfortsetzung.

2.2.9.3. Fehlerformen

Durch die im Prüfprogramm angegebene TRAP-Kodierung können 8 verschiedene Formen für die Fehlerinformationsausgabe angewiesen werden.
Folgende Formen der Fehlerinformationsausgabe sind vereinbart:

Form	Fehlerinformation	Verwendung
0	R0, R1, R2, R3, R4, R5 SZ, SV, SP, PS, PC, IV	Allgemeine Form, wenn keine Informationszuordnung möglich ist
1	AD/BC, IS, SL, PS, PC, IV	Ist/Sollwertvergleich Die Modifizierung AD/BC erfolgt in Abhängigkeit von Fehlerkennzeichen
2	AD, SW, IS, SL, PC, IV	Ist/Sollwertvergleich mit Status/Steuer
3	SW, SA, IS, SL, SV, PC	Speichertestfehlerausgabe SW enthält Hinweise zur Fehlerart
4	AS, SW, C1, C2, C3, C4 C5, C6, C7, C8, PC, IV	Fehlerausgabe für Geräte/AS mit mehr als einem Statusregister
5	AD, SC, X1, X2 ... X9 PC IV	Serieller Status X = ASCII-Statuszeichen
6	AD SW AD SW PC IV D1 D2 D3 D4 D5 D8	Doppeladresse mit Status Datenwörter in Fehlerbereich
7	PC Fehlertext max. 38 ASCII-Zeichen	Fehlertextausgabe, wenn keine oktale Informationszuordnung möglich

Alle Ziffernausgaben in den Fehlerformen erfolgen oktal. Die Bedeutung ist den vorausstehenden Kurzzeichen zu entnehmen. Ist die dadurch gegebene Kennzeichnung zu allgemein, wird auf die Beschreibung der die Fehlerausschrift verursachenden Prüfprogramme verwiesen.

2.2.10. Anfangsladen des LPROEM und Bedienung der Folienspeichereinheit des Konsolrechners

Das Leitprogramm LPROEM ist speziell für den externen Datenträger Folienspeichereinheit ausgelegt und wird von dort über die Diskette DS0 geladen.

2.2.10.1. Start über Anfangslader

Die Diskette mit der PSU des Konsolrechners (DS0) ist in ein Konsollaufwerk zu geben. Über Rechnerstart oder 173000G im Bedienzustand des Konsolrechners wird der Anfangslader für die FSE gestartet. Bei richtigem Lesevorgang meldet sich LPROEM entsprechend 2.2.10.2..
Bei fehlerhaftem Anfangsladen (Halt auf Adresse 2, Halt auf Adresse 10 oder Endlosschleife des Anfangsladers) ist das Register R6 zu kontrollieren und auf R6 > 400 einzustellen, die Diskette bzw. die FSE-Konfiguration zu überprüfen und der Anfangsladerstart zu wiederholen.

2.2.10.2. Gerätebedienung FSE

Vor Start des LPROEM über Anfangslader Folienspeicher bzw. über 20000G aus dem Hauptspeicher ist die Diskette DS0 in ein Laufwerk der Konsol-FSE einzulegen. LPROEM meldet sich auf dem Bediengerät mit den Systemzeilen:

```
LPROEM        KSS  VERS.  X-87
BEDIENUNG     KMBE [KS] - FSE [FS]
ZE K1620
OPERATIVSPEICHER =        000000 137776
PSU-SPEICHERBEREICH       065400 137776;
```

Da keine Änderung des PSU-Speicherbereiches vorgenommen wird, gibt man "<CR>". Es erscheint die Anfrage zur

GERAETEKONFIGURATION:

Diese ist im Falle des Konsolrechners mit "FS <CR>" zu beantworten. Die Laufwerksanfrage

DISKETTE LAUFWERK (0\1);

ist mit der entsprechenden Laufwerksnummer (Standard: DS0 in LW0 —> 0) zu quittieren. Dadurch wird

- die FSE initialisiert,
- die Laufwerksbereitschaft geprüft,
- das Inhaltsverzeichnis geladen.

Die erfolgreiche Beendigung dieses Ablaufes wird mit ":" angezeigt. Der LPROEM-Start ist abgeschlossen; durch Eingabe von <CR> wird die Bedienungsanleitung bzw. das Promptzeichen ">" angeboten (je nach Startart).
War die FSE zwischenzeitlich ausgeschaltet, ist LPROEM im Bedienmodus des Konsolrechners über 20000G (Neustart) oder 20026G (Wiederstart) zu starten, da dadurch die Grundeinstellung der Folienspeichereinheit vorgenommen wird.

2.2.10.3. Programmtechnische Realisierung

Die Aufzeichnung der Prüfprogramme auf Diskette DS0 erfolgt ab Spur 14 in fortlaufender Reihenfolge.
Für jedes Prüfprogramm werden Spurnummer, Sektornummer, Sektoranzahl, Gültigkeitskennzeichen, Programmkennzeichen, Prüfsumme in entsprechenden Bibliothekstabellen gespeichert. Somit ist die Prüfprogrammkonfiguration immer verfügbar und jedes Programm durch direkte Adressierung sofort lesbar, d. h. ein Einfluß von Aufzeichnungsreihenfolge und Abarbeitungsreihenfolge auf die Abarbeitungsgeschwindigkeit besteht nicht.
LPROEM arbeitet nur mit dem angewählten Laufwerk. Aufzuzeichnende Dauertestgenerierungen werden über DGNAUS auf die angewählte Diskette aufgezeichnet.
Bei Ausgabe des Disketteninhalts werden nur die enthaltenen Prüfprogramme mit Programmlänge und Ausgabenummer angezeigt. Zusätzlich wird die Belegung der Diskette (Spur- und Sektornummer) oktal ausgegeben.

2.3. Beschreibung des Prüfprogramms ZVZW20

2.3.1. Benennung

Test der Verzweigungen und der Einstellbefehle für die Bedingungsbits

2.3.2. Dokumentationsnummer

Die Dokumentationsnummer für das auf der Diskette DS0 geladene
Prüfprogramm ZVZW20 lautet 1.57.700201.1 ,
für das dazugehörige Prüfprogramm 1.57.700226.1 .

2.3.3. Programmumfang

Das Programm enthält die folgenden 14 Routinen zum Test der Befehlsabläufe bezüglich der ermittelten Verzweigungsadressen.

Routinenummer	Routinename	Routinefunktion
RT01	BCCN0	BCC-Test CLC-Test
RT02	BCSN0	BCS-Test SEC-Test
RT03	BVCN0	BVC-Test CLV-Test
RT04	BVSN0	BVS-Test SEV-Test
RT05	BNEN0	BNE-Test CLZ-Test
RT06	BEQN0	BEQ-Test SEZ-Test
RT07	BPLN0	BPL-Test

		CLN-Test
RT10	BMIN0	BMI-Test SEN-Test
RT11	BGEN0	BGE-Test
RT12	BLTN0	BLT-Test
RT13	BGTN0	BGT-Test
RT14	BLEN0	BLE-Test
RT15	BHIN0	BHI-Test
RT16	BLOSN0	BLOS-Test

2.3.4. Speicherbedarf

000000...002777

2.3.5. Ausführungszeit

ca. 2 sec.

2.3.6. Testhilfsmittel/Geräteeinstellungen

entfällt

2.3.7. Arbeit unter LPROEM

Aufruf: Eingabe des Namens ZVZW20 (s. LPROEM)
Generierung: (s. Pkt. 2.3.9. und LPROEM)
Fehlerausgabe:(s. Pkt. 2.3.10.)

2.3.8. Arbeit ohne LPROEM

Start: Programmanfangsadresse +14
Generierhalt: Programmanfangsadresse +22 (s. Pkt. 2.3.9.)
Fehlerart: Programmanfangsadresse +74 (s. Pkt. 2.3.10.)
Programmende: Programmanfangsadresse +62

Bei Interrupt geht das Programm auf der jeweiligen Vektoradresse +4 in den HALT-Zustand.

2.3.9. Generierinformationen

Die Eintragungen werden auf folgenden Zeilen vorgenommen:
(Standardfall in Klammer)

7002 PSW : (000000)
Eintragung der gewünschten Prozessorpriorität

7004 Teststeuerwort : (100002)
bit1 = 1 (Einzeltest)

7006 Routinesteuerwort : (037777)
Ausblenden beliebiger Routinen der vorgegebenen maximalen Routinezahl
z.B. 37777 wird in 37577 geändert. Ausblenden der 8. Routine.

7012 Zykluszähler Prüfprogramm : (000001)
000001 ... 177777

7014 Zykluszähler Routine : (000001)
000001 ... 177777

Die Eintragungen erfolgen bei Arbeitsweise mit LPROEM im Kommandoeingabezustand des LPROEM (s. LPROEM) bzw. bei Arbeitsweise ohne LPROEM im Bedienzustand des Konsolrechners.

2.3.10. Fehlerinterpretation

2.3.10.1. Arbeitsweise mit LPROEM

Im Fehlerfalle erfolgt die Ausgabe von notwendigen Informationen zur Fehlersuche in folgender Form:

BF ZVZW20 RTXX BC IS SL PS PC

BF	Befehlsfehler
ZVZW20	Programmname
RTXX	Routinenummer
BC	Befehlscode
IS	Istwert
SL	Sollwert
PS	PSW des Fehler-TRAP
PC	PC des Fehler-TRAP

Die Bedeutung anderer Fehlerausschriften ist im Pkt. 2.2.9.2. erläutert.

2.3.10.2. Arbeitsweise ohne LPROEM

Im Fehlerfalle können den Registern folgende Fehlerinformationen entnommen werden:

R0	Befehlscodierung
R4	PC des Fehler-TRAP

2.3.10.3. Funktionsbeschreibung

Der Test der einfachen Verzweigebefehle erfolgt durch Vorgabe der Verzweigungsbedingungen im PSW und nachfolgender Durchführung des jeweils zugehörigen Verzweigungsbefehls. Ein so getesteter Verzweigebefehl kann im nachfolgenden Text der Bedingungsbitbefehle als funktionstüchtig vorausgesetzt werden. Zum Test der kombinierten Verzweigebefehle können nunmehr die Bedingungsbit-Befehle zur Vorgabe der Verzweigungsbedingungen verwendet werden.
Die Tests erfolgen jeweils für erfüllte und für nicht erfüllte Verzweigungsbedingungen.
Der Test ist in 14 Routinen unterteilt.

Jede Routine entspricht entweder dem Test eines Verzweigebefehls (kombinierte Verzweigebefehle) oder aber dem Test eines Verzweigebefehls und des zugehörigen Bedingungsbit-Befehls (einfache Verzweigebefehle).

2.4. Beschreibung des Prüfprogramms ZARI20

2.4.1. Benennung

Test der Arithmetikbefehle

2.4.2. Dokumentationsnummer

Das auf der Diskette DS0 geladene Prüfprogramm ZARI20 hat die
Dokumentationsnummer 1.57.700202.8 ,
die dazugehörige Programmliste 1.57.700227.8

2.4.3. Programmumfang

Das Programm enthält die folgenden 10 Routinen zum Test der Befehlsabläufe bezüglich ihrer Wirkung auf die am Befehlsablauf beteiligten Operanden.

Routinenummer	Routinename	Routinefunktion
RT01	ADDR	ADD-Test SUB-Test
RT02	NEGR	NEG-Test NEGB-Test
RT03	COMR	COM-Test COMB-Test
RT04	INCR	INC-Test INCB-Test
RT05	DECR	DEC-Test DECB-Test
RT06	ADCR	ADC-Test ADCB-Test
RT07	SBCR	SBC-Test SBCB-Test
RT10	SWABR	SWAB-Test
RT11	BICR	BIC-Test BICB-Test
RT12	BISR	BIS-Test BISB-Test

2.4.4. Speicherbedarf

000000...001777

2.4.5. Ausführungszeit

ca. 2 min.

Testhilfsmittel/Geräteeinstellungen

entfällt

2.4.6. Arbeit unter LPROEM

Aufruf: Eingabe des Namens ZARI20 (s. LPROEM)
Generierung: (s. Pkt. 2.4.9. und LPROEM)
Fehlerausgabe: (s. Pkt. 2.4.10.)

2.4.7. Arbeit ohne LPROEM

Start: Programmanfangsadresse +14
Generierhalt: Programmanfangsadresse +22 (s. Pkt. 2.4.9.)
Fehlerhalt: Programmanfangsadresse +74 (s. Pkt. 2.4.10.)
Programmende: Programmanfangsadresse +62

Bei Interrupt geht das Programm auf der jeweiligen Vektoradresse +4 in den HALT-Zustand.

2.4.8. Generierinformationen

Die Eintragungen werden auf folgenden Zellen vorgenommen:
(Standardfall in Klammer)

7002 PSW : (000000)
Eintragung der gewünschten Prozessorpriorität

7004 Teststeuerwort : (000002)
keine Veränderung erforderlich

7006 Routinesteuerwort : (001777)
Ausblenden beliebiger Routinen aus der vorgegebenen maximalen Routinezahl.
z.B. 1777 wird in 1377 geändert, d.h. die 8. Routine, der Test des SWAB-Befehls, wird ausgeblendet.

7012 Zykluszähler Prüfprogramm : (000001)
000001...177777

7014 Zykluszähler Routine : (000001)
000001...177777

Die Eintragungen erfolgen bei Arbeitsweise mit LPROEM im Kommandoeingabezustand

des LPROEM (s. LPROEM) bzw. bei Arbeitsweise ohne LPROEM im Bedienzustand des Konsolrechners.

2.4.9. Fehlerinterpretation

2.4.9.1. Arbeitsweise mit LPROEM

Im Fehlerfall erfolgt die Ausgabe von notwendigen Informationen zur Fehlersuche in folgender Form:

BF ZARI20 RTXX BC IS SL PS PC

BF	Befehlsfehler
ZARI20	Programmname
RTXX	Routinenummer
BC	Befehlscode
IS	Istwert des Operanden
SL	Sollwert des Operanden
PS	PSW des Fehler-TRAP
PC	PC des Fehler-TRAP

Die Bedeutung anderer Fehlerausschriften ist im Pkt. 2.2.9.2. erläutert.

2.4.9.2. Arbeitsweise ohne LPROEM

Im Fehlerfalle können den Registern folgende Informationen entnommen werden:

RO	Befehlscodierung
R1	Istwert des Operanden
R2	Sollwert des Operanden
R4	PC des Fehler-TRAP

2.4.10. Funktionsbeschreibung

Der Test erfolgt durch Simulation der Sollwerte. Zu diesem Zweck ist der Test bezüglich der Routinefolge so aufgebaut, daß die jeweils getesteten Befehlsabläufe in der folgenden Routine als funktionstüchtig vorausgesetzt und zur Simulation der Sollwerte verwendet werden.
Dieses Prinzip der Sollwertbereitstellung ermöglicht die Einbeziehung aller Operanden des Wertebereiches 000000... 177777 in den Test und somit einen umfassenden Nachweis der Funktionstüchtigkeit.
Um auch ohne Voraussetzung wesentlicher anderer Befehle zu Beginn des Testprogramms das Prinzip der Simulation einzuhalten, wurden die Befehle ADD und SUB in einer Routine zusammengefaßt. Die Kombination dieser beiden Befehle läßt wiederum die Einbeziehung aller Operanden in den Test zu. In allen anderen Fällen wird jeweils der betreffende Wort-und Bytebefehl zu einer Routine zusammengefaßt.

2.5. Beschreibung des Prüfprogramms ZSHI20

2.5.1. Benennung

Test der Shiftbefehle

2.5.2. Dokumentationsnummer

Das auf der Diskette DS0 geladene Prüfprogramm ZSHI20 hat die
Dokumentationsnummer 1.57.700203.6 ,
die dazugehörige Programmliste 1.57.700228.6 .

2.5.3. Programmumfang

Das Programm enthält die folgenden 4 Routinen zum Test der Befehlsabläufe bezüglich ihrer Wirkung auf die zu verschiebenden Operanden:

Routinenummer	Routinename	Routinefunktion
RT01	ROLR	ROL-Test ROLB-Test
RT02	RORR	ROR-Test RORB-Test
RT03	ASLR	ASL-Test ASLB-Test
RT04	ASRR	ASR-Test ASRB-Test

2.5.4. Speicherbedarf

000000...001777

2.5.5. Ausführungszeit

ca. 5 sec.

2.5.6. Testhilfsmittel/Geräteeinstellungen

entfällt

2.5.7. Arbeit unter LPROEM

Aufruf: Eingabe des Namens ZSHI20 (s. LPROEM)
Generierung: (s. Pkt. 2.5.9. und LPROEM)
Fehlerausgabe: (s. Pkt. 2.5.10.)

2.5.8. Arbeit ohne LPROEM

Start: Programmanfangsadresse +14
Generierhalt: Programmanfangsadresse +22 (s. Pkt. 2.5.9.)
Fehlerhalt: Programmanfangsadresse +74 (s. Pkt. 2.5.10.)
Programmende: Programmanfangsadresse +62

Bei Interrupt geht das Programm auf der jeweiligen Vektoradresse +4 in den HALT-Zustand.

2.5.9. Generierinformationen

Die Eintragungen werden auf folgenden Zellen vorgenommen:
(Standardfall in Klammer)

7002 PSW : (000000)
Eintragung der gewünschten Prozessorpriorität

7004 Teststeuerwort : (000002)
keine Veränderung erforderlich

7006 Routinesteuerwort : (000017)
Ausblenden beliebiger Routinen aus der vorgegebenen maximalen Routinezahl z.B. 17 wird in 10 geändert, d.h. es erfolgt nur der Test für die Befehle ROL und ROLB

7012 Zykluszähler Prüfprogramm (000001)
000001...177777

7014 Zykluszähler Routine (000001)
000001...177777

Die Eintragungen erfolgen bei Arbeitsweise mit LPROEM im Bedienzustand des Konsolrechners.

2.5.10. Fehlerinterpretation

2.5.10.1. Arbeitsweise mit LPROEM

Im Fehlerfall erfolgt die Ausgabe von notwendigen Informationen zur Fehlersuche in folgender Form:

BF ZSHI20 RTXX BC IS SL PS PC

BF	Befehlsfehler
ZSHI20	Programmname
RTXX	Routinenummer
BC	Befehlscode
IS	Istwert des Operanden
SL	Sollwert des Operanden
PS	PSW des Fehler-TRAP
PC	PC des Fehler-TRAP

Die Bedeutung anderer Fehlerausschriften ist im Pkt. 2.2.9.2. erläutert.

2.5.10.2. Arbeitsweise ohne LPROEM

Im Fehlerfall können den Registern folgende Informationen entnommen werden:

R0 Befehlscodierung
R1 Istwert des Operanden
R2 Sollwert des Operanden
R4 PC des Fehler-TRAP

2.5.11. Funktionsbeschreibung

Der Test erfolgt durch Simulation der Sollwerte und Vergleich derselben mit den durch Befehlsabarbeitung erzielten Istwerten. Diese Simulation gestattet durch Verschiebung den Durchlauf einer Bitstelle durch das gesamte Wort bzw. Byte und somit einen ausführlichen Test.
Die 4 Routinen enthalten jeweils den Wort- und Bytebefehl einer Verschiebeoperation in einer Routine zusammengefaßt.

2.6. Beschreibung des Prüfprogramms ZBBI20

2.6.1. Benennung

Test der Bedienungsbits bezüglich ihrer Einstellung durch die Standardbefehle des Konsolrechners

2.6.2. Dokumentationsnummer

Das auf der Diskette DS0 geladene Prüfprogramm ZBBI20 hat die
Dokumentationsnummer 1.57.700204.4 ,
die dazugehörige Programmliste 1.57.700229.4 .

2.6.3. Programmumfang

Das Programm enthält die folgenden 16 Routinen zum Test der Befehlsabläufe bezüglich ihrer Wirkung auf die Bedingungsbits.

Routinenummer	Routinename	Routinefunktion
RT01	COMZ	COM-Test
RT02	INCZ	INC-Test
	DECZ	DEC-Test
RT03	NEGZ	NEG-Test
RT04	TSTZ	TST-Test
RT05	RORZ	ROR-Test
	ROLZ	ROL-Test
	ASRZ	ASR-Test
	ASLZ	ASL-Test
RT06	SWABZ	SWAB-Test

RT07	ADCZ	ADC-Test
	SBCZ	SBC-Test
RT10	ADDZ	ADD-Test
	SUBZ	SUB-Test
RT11	BITZ	BIT-Test
	BICZ	BIC-Test
	BISZ	BIS-Test
RT12	COMBZ	COMB-Test
RT13	INCBZ	INCB-Test
	DECBZ	DECB-Test
RT14	NEGBZ	NEGB-Test
RT16	TSTBZ	TSTB-Test
RT17	ADCBZ	ADCB-Test
	SBCBZ	SBCB-Test
RT20	BITBZ	BITB-Test
	BICBZ	BICB-Test
	BISBZ	BISB-Test

2.6.4. Speicherbedarf

000000...004777

2.6.5. Ausführungszeit

ca. 2 sec.

2.6.6. Testhilfsmittel/Geräteeinstellungen

entfällt

2.6.7. Arbeit unter LPROEM

Aufruf: Eingabe des Namens ZBBI20 (s. LPROEM)
Generierung: (s. Pkt. 2.6.9. und LPROEM)
Fehlerausgabe: (s. Pkt. 2.6.10.)

2.6.8. Arbeit ohne LPROEM

Start: Programmanfangsadresse +14
Generierhalt: Programmanfangsadresse +22 (s. Pkt. 2.6.9.)
Fehlerhalt: Programmanfangsadresse +74 (s. Pkt. 2.6.10.)
Programmende: Programmanfangsadresse +62

Bei Interrupt geht das Programm auf der jeweiligen Vektoradresse +4 in den HALT-Zustand.

2.6.9. Generierinformationen

Die Eintragungen werden auf folgenden Adressen vorgenommen (Standardfall in Klammer):

7002 PSW : (000000)
Eintragung der gewünschten Priorität für das Prüfprogramm

7004 Teststeuerwort : (100002)
bit1 = 1 (Einzeltest)

7006 Routinesteuerwort : (177777)
Ausblenden beliebiger Routinen aus der vorgegebenen maximalen Routinezahl z.B. 177777 wird in 177752 geändert, d.h. die 1., 3. und 5. Routine wird ausgeblendet.

7012 Zykluszähler Prüfprogramm : (000001)
000001...177777

7014 Zykluszähler Routine (000001)
000001...177777.

Die Eintragungen erfolgen bei Arbeitsweise mit LPROEM im Kommandoeingabezustand des LPROEM (s. LPROEM) bzw. bei Arbeitsweise ohne LPROEM im Bedienzustand des Konsolrechners.

2.6.10. Fehlerinterpretation

2.6.10.1. Arbeitsweise mit LPROEM

Im Fehlerfalle erfolgt die Ausgabe von notwendigen Informationen zur Fehlersuche in folgender Form:

BF ZBBI20 RTXX BC IS SL PS PC

BF	Befehlsfehler
ZBBI20	Programmname
RTXX	Routinenummer
BC	Befehlscode
IS	Istwert der Bedingungsbits
SL	Sollwert der Bedingungsbits
PS	PSW des Fehler-TRAP
PC	PC des Fehler-TRAP

Die Bedeutung anderer Fehlerausschriften ist im Pkt. 2.2.9.2. erläutert.

2.6.10.2. Arbeitsweise ohne LPROEM

Im Fehlerfall können den Registern folgende Informationen entnommen werden:

R0	Befehlscodierung
R1	Istwert der Bedingungsbits
R2	Sollwert der Bedingungsbits
R4	PC des Fehler-TRAP

2.6.11. Funktionsbeschreibung

Es werden jeweils die Sollinformationen für die Bedingungsbits sowie die entsprechenden Ausgangsinformationen für den gewünschten Befehlsablauf bereitgestellt.
Nach Durchführung des zu testenden Befehls erfolgt der Vergleich mit den Bedingungsbits des Zustandswortes.
Jeder unter Pkt. 2.6.3. aufgeführte Test ist im Prinzip ein selbständiges abgeschlossenes Programm und könnte als Routine aufrufbar sein. Da jedoch maximal 16 Routinen vom Leitprogramm zugelassen sind, wurden jeweils die Programme zu einer Routine zusammengefaßt, die den Test analoger Befehle zum Inhalt haben (z.B. Verschiebebefehle oder ADC und SBC usw.).

2.7. Beschreibung des Prüfprogramms ZBEF30

2.7.1. Benennung

Befehlstest des Konsolrechners

2.7.2. Dokumentationsnummer

Das auf der Diskette DS0 geladene Prüfprogramm ZBEF30 hat die
Dokumentationsnummer 1.57.700205.2 ,
die dazugehörige Programmliste 1.57.700230.0 .

2.7.3. Programmumfang

Das Programm enthält die folgenden 15 Routinen für den Test der Befehle:
MUL, DIV, BMOV, XOR, SXT, SOB, ASH, ASHC, MARK und SPL

Routinenummer	Routinename	Routinefunktion
RT01	ADRT	Test der Adressierungsmodi für die gesamten Befehle (Prinziptest)
RT02	MULIR	MUL-Test (Multiplikator indirekt adressiert, Multiplikand und Produkt im Registermodus)
RT03	DIVIR	DIV-Test (Divisor indirekt adressiert, Dividend und Quotient im Registermodus)
RT04	BMOVS	BMOV-Test (steigende Adressen)
RT05	BMOVF	BMOV-Test (fallende Adressen)
RT06	XORR	XOR-Test (Registermodus)
RT07	SXTR	SXT-Test (Registermodus)

RT10	SOBN	SOB-Test (Test für nicht erfüllte und erfüllte (SOB)BRANCH-Bedingung)
RT11	ASHL	ASH-Test (Linksverschiebung)
RT12	ASHR	ASH-Test (Rechtsverschiebung)
RT13	ASHCL	ASHC-Test (Linksverschiebung, geradzahliges Register, d.h. Doppelwortverschiebung von R(n+1) nach Rn)
RT14	ASHCR	ASHC-Test (Rechtsverschiebung, geradzahliges Register, d.h. Doppelwortverschiebung von Rn nach R(n+1))
RT15	MARK	MARK-Test
RT16	ZBBI30	Test der Wirkung genannter Befehle auf die Zustandsbits XORZ SXTZ ASHZ ASHCZ (ungeradzahliges Register, d.h. zyklischer Umlauf des Operanden in R(n+1)) BMOVZ MULZ DIVZ (Besonderheit: Bei Division mit Null bleiben die Bedingungsbits N und Z entsprechend dem vorangegangenen Befehl erhalten).
RT17	SPL	SPL-Test (Besonderheit: Dieser Test verändert das Prioritätsniveau, liegt deshalb aus Sicherheitsgründen am Schluß des Testprogrammes).

2.7.4. Speicherbedarf

000000... 005777

2.7.5. Ausführungszeit

ca. 500 sec.

2.7.6. Testhilfsmittel/Geräteeinstellungen

entfällt

2.7.7. Arbeit unter LPROEM

Aufruf: Eingabe des Namens ZBEF30 (s. LPROEM)
Generierung: (s. Pkt. 2.7.9. und LPROEM)
Fehlerausgabe: (s. Pkt. 2.7.10.)

2.7.8. Arbeit ohne LPROEM

Start: Programmanfangsadresse +14
Generierhalt: Programmanfangsadresse +22 (s. Pkt. 2.7.9.)
Fehlerhalt: Programmanfangsadresse +74 (s. Pkt. 2.7.10.)
Programmende: Programmanfangsadresse +62

Bei Interrupt geht das Programm auf der jeweiligen Vektoradresse +4 in den HALT-Zustand.

2.7.9. Generierinformationen

Die Eintragungen werden auf folgenden Zellen vorgenommen:
(Standardfall in Klammer)

7002 PSW: (000000)
Eintragung der gewünschten Prozessorpriorität

7004 Teststeuerwort: (100002)
Bit1 = 1 (Einzeltest)

7006 Routinesteuerwort: (077777)
Ausblenden beliebiger Routinen der vorgegebenen maximalen Routinezahl
z.B. 77777 wird in 77770 geändert, d.h. Ausblenden der
1., 2. und 3. Routine

7012 Zykluszähler Prüfprogramm: (000001)
000001...177777

7014 Zykluszähler Routine: (000001)
000001...177777

Die Eintragungen erfolgen bei Arbeitsweise mit LPROEM im Kommandoeingabezustand des LPROEM (s. LPROEM) bzw. bei Arbeitsweise ohne LPROEM im Bedienzustand des Konsolrechners.

2.7.10. Fehlerinterpretation

2.7.10.1. Arbeitsweise mit LPROEM

Im Fehlerfalle erfolgt die Ausgabe von notwendigen Informationen zur Fehlersuche in folgender Form:

Routinen RT01, RT04...RT17

```
BF  ZBEF30 RTXX BC IS SL PS PC
  BF       Befehlszähler
  ZBEF30   Programmname
  RTXX     Routinenummer
```

BC	Befehlscode
IS	Istwert (bei RT10 ohne Bedeutung)
SL	Sollwert (bei RT10 ohne Bedeutung)
PS	PSW des Fehler-TRAP
PC	PC des Fehler-TRAP

Routinen RT02, RT03

BF ZBEF30 RTXX R0 R1 R2 R3 R4 R5
SP PS PC

BF	Befehlszähler
ZBEF30	Programmname
RTXX	Routinenummer
R0	Befehlscode
R1	(ohne Bedeutung)
R2	Sollwert höherwertiger Teil
R3	Sollwert niederwertiger Teil
R4	Istwert höherwertiger Teil
R5	Istwert niederwertiger Teil
SP	Stack
PS	PSW des Fehler-TRAP
PC	PC des Fehler-TRAP

Die Bedeutung anderer Fehlerausschriften ist im Pkt.2.2.9.2. erläutert.

2.7.10.2. Arbeitsweise ohne LPROEM

Im Fehlerfalle können den Registern folgende Informationen entnommen werden:

Routinen RT01, RT04...RT17

R0	**Befehlscode**
R1	**Istwert (außer bei RT10)**
R2	**Sollwert (außer bei RT10)**
R4	**PC des Fehler-TRAP**

Routinen RT02, RT03

R0	**Befehlscode**
R2	**Sollwert höherwertiger Teil**
R3	**Sollwert niederwertiger Teil**
R4	**PC des Fehler-TRAP**
R5	**Istwert niederwertiger Teil**
R6	**Stackadresse (enthält den Istwert höherwertiger Teil)**

2.7.11. Funktionsbeschreibung

RT01: Test aller Adressierungsarten der im Programm getesteten Befehle in Form eines Kurztests. Im allgemeinen ist der Test so aufgebaut, daß der Befehl 8mal jeweils mit anderem Adreßmodus abgearbeitet, auf das Ergebnis des vorangegangenen Befehls wirkt und das endgültige Ergebnis dem Ausgangsoperanden oder einem vorgegebenen Wert entspricht. Tritt ein Fehler auf, so ist es zweckmäßig, die Routine im Schrittbetrieb abzuarbeiten, um den fehlerhaft arbeitenden Adreßmodus zu ermitteln.

RT02: Der Test erfolgt auf der Basis der Simulation mittels der Doppelwortverschiebung um 0, 1, 2, 3...16 Bit. Dies entspricht der Multiplikation mit diskreten Multiplikatoren. Hierbei müssen bestimmte Randbedingungen berücksichtigt werden und Vorzeichenfunktionen des Multiplikators in geeigneter Form in den Multiplikanden übernommen werden.

- Multiplikator = 0: Löschen des Multiplikanden
- Multiplikator negativ, Multiplikand positiv: Vorzeichenerweiterung
- Multiplikator positiv, Multiplikand negativ: Übernahme des negativen Vorzeichens durch Negieren des Multiplikanden und Vorzeichenerweiterung
- Multiplikator negativ, Multiplikand negativ: Übernahme des positiven Vorzeichens des Produkts durch Negieren des Multiplikanden

Die Verschiebungen entsprechen den Multiplikatoren 1, 2, 4, 10, ...10000, 20000, 40000 bzw. 177777, 177776, 177774...170000, 160000, 140000.
Der Multiplikand kann alle Werte zwischen 100000 und 077777 durchlaufen, aus Gründen der Laufzeit des Testprogramms ist der Test jedoch auf die Werte 0, 10000, 20000, 30000, ...100000, ...150000, 160000, 170000 beschränkt.

RT03: Test durch Kombination mit der Multiplikation, so daß der als Multiplikand bereitgestellte Ausgangsoperand mit dem Quotienten der Division identisch ist und als Sollwert zur Verfügung steht. Aus Gründen der Laufzeit des Testprogramms erfolgte wiederum Beschränkung auf die Operanden 0, 10000, 20000, 30000, ...100000, ...150000, 160000, 170000.

RT04: Transport eines Feldes von 10 Worten nach Adressen oberhalb des Quellfeldes und sich mit diesen überlappend.

RT05: Transport eines Feldes von 10 Worten nach Adressen unterhalb des Quellfeldes und sich mit diesen überlappend.

RT06: Test mit allen Modifikationen beider Operanden, die zum (XOR-)Ergebnis 0 bzw. 177777 führen (2 hoch 16 Tests).

RT07: Test durch Vorgabe des N-Bit (2 hoch 16 Tests)

RT10: Test durch Vorgabe der Verzweigungsbedingung (erfüllt und nicht erfüllt) (2 hoch 16 Tests)

RT11/12: Test auf der Basis der Simulation mittels der einfachen Verschiebung. Die Information durchläuft alle Bitstellen, so daß der Verschiebezähler von 0 bis 15 getestet wird (2 x 16 Tests)

RT13/14: Test auf der Basis der Simulation mittels der einfachen Verschiebung. Die Information durchläuft alle Bitstellen, so daß der Verschiebezähler von 0 bis 31 getestet wird.

RT15: Test mit bereitgestellten Sollwerten.

RT16: Test mit vorgegebenen Operanden, die zur geforderten Einstellung der Zustandsbits führen.

RT17: Test durch Vorgabe des Prioritätslevels und Auswertung des Statuswortes PSW.

2.8. Beschreibung des Prüfprogramms ZADRAR

2.8.1. Benennung

Test der Adressierungsarten und PSW-Register-Befehle

2.8.2. Dokumentationsnummer

Das auf die Diskette DS0 geladene Prüfprogramm ZADRAR hat die
Dokumentationsnummer 1.57.700206.0 ,
die dazugehörige Programmliste 1.57.700231.7

2.8.3. Programmumfang

Das Programm enthält die folgenden 5 Routinen zum Test der Adressierungsarten bzw. der Befehle MFPS und MTPS.

Routinenummer	Routinename	Routinefunktion
RT01	ADR1T	Test des Adreßmodus 2 Test des Adreßmodus 4
RT02	ADR2T	Test des Adreßmodus 3 Test des Adreßmodus 5
RT03	ADR3T	Test des Adreßmodus 6
RT04	ADR4T	Test des Adreßmodus 7
RT05	MPST	MFPS-Test MTPS-Test

2.8.4. Speicherbedarf

000000...001777

2.8.5. Ausführungszeit

ca. 30 sec.

2.8.6. Testhilfsmittel/Geräteeinstellungen

entfällt

2.8.7. Arbeit unter LPROEM

Aufruf: Eingabe des Namens ZADRAR (s. LPROEM)
Generierung: (s. Pkt. 2.8.9. und LPROEM)
Fehlerausgabe: (s. Pkt. 2.8.10.)

2.8.8. Arbeit ohne LPROEM

Start: Programmanfangsadresse +14
Generierhalt: Programmanfangsadresse +22 (s. Pkt. 2.8.9.)
Fehlerhalt: Programmanfangsadresse +74 (s. Pkt. 2.8.10.)
Programmende: Programmanfangsadresse +62

Bei Interrupt geht das Programm auf der jeweiligen Vektoradresse +4 in den HALT-Zustand.

2.8.9. Generierinformationen

Die Eintragungen werden auf folgenden Adressen vorgenommen:
(Standardfall in Klammer)

7002 PSW : (000000)
Eintragung der gewünschten Prozessorpriorität für das Prüfprogramm

7004 Teststeuerwort : (000002)
keine Veränderung erforderlich

7006 Routinesteuerwort : (000037)
Ausblenden beliebiger Routinen aus der vorgegebenen maximalen Routinezahl.

7012 Zykluszähler Prüfprogramm : (000001)
000001...177777

7014 Zykluszähler Routine : (000001)
000001...177777

Die Eintragungen erfolgen bei Arbeitsweise mit LPROEM im Kommandoeingabezustand des LPROEM (s. LPROEM) bzw. bei Arbeitsweise ohne LPROEM im Bedienzustand des Konsolrechners.

2.8.10. Fehlerinterpretation

2.8.10.1. Arbeitsweise mit LPROEM

Im Fehlerfalle erfolgt die Ausgabe von notwendigen Informationen zur Fehlersuche in folgender Form:

BF ZADRAR RTXX BC IS SL PS PC

BF	Befehlsfehler
ZADRAR	Programmname
RTXX	Routinenummer
BC	Adreßmodi (OOOOXY : Modus X und Y)
IS	Istwert des Operanden
SL	Sollwert des Operanden
PS	PSW des Fehler-TRAP
PC	PC des Fehler-TRAP

Die Bedeutung anderer Fehlerausschriften ist im Pkt. 2.2.9.2. erläutert.

2.8.10.2. Arbeitsweise ohne LPROEM

Im Fehlerfalle können den Registern folgende Informationen entnommen werden:

R0 Adreßmodi
R1 Istwert des Operanden
R2 Sollwert des Operanden
R4 PC des Fehler-TRAP

2.8.11. Funktionsbeschreibung

Der Test ist auf dem Befehl MOV aufgebaut. Es erfolgt somit der Funktionsnachweis der Adreßrechnung für diesen Befehl in vollem Umfang. Für alle anderen Befehle ist der Funktionsnachweis der Adreßrechnung in zentralen Logikbereichen erbracht. Ein umfassender Test, der die Adreßrechnung eines jeden Mikroprogramms bezüglich aller Adreßmodi durchführt, liegt infolge des außergewöhnlich hohen Aufwands explizit nicht vor, ist aber nahezu vollständig im gesamten Befehlstest implizit vorhanden. Zum Test wurden jeweils Lese- und Schreibvorgänge kombiniert, so daß das Testprinzip, die Bereitstellung von Sollwerten auch hier beibehalten werden konnte. Die Routine 5 testet die Transportbefehle zu und vom Prozessorstatuswort ebenfalls durch Kombination beider Befehle MFPS und MTPS, so daß bis auf die T-Bit-Stelle alle 16 Bitstellen des PSW getestet werden. Ein Test der T-Bit-Stelle würde zur Unterbrechung des Programms führen, er konnte hier unberücksichtigt bleiben, da er im Interrupttest voll enthalten ist.
Es sind jeweils die Adreßmodi 2 und 4, 3 und 5 bzw. die Befehle MFPS und MTPS zu je einer Routine zusammengefaßt.

2.9. Beschreibung des Prüfprogrammes ZSPE20

2.9.1. Benennung

Test aller vom Programm und Bediengerät beschreibbaren Speicherarten des Konsolrechners

2.9.2. Dokumentationsnummer

Das auf der Diskette DS0 geladene Prüfprogramm ZSPE20 hat die
Dokumentationsnummer 1.57.700207.7 ,
die dazugehörige Programmliste 1.57.700232.5 .

2.9.3. Programmumfang

Routinenummer	Routinename	Routinefunktion
RT01	MART	Vorwärts/rückwärts marschierende 1/0 (Marching-Test)
RT02	REGT	Regeneriertest

2.9.4. Speicherbedarf

000000...002777

2.9.5. Ausführungszeit

Die Ausführungszeit ist abhängig von der Größe der zu testenden Speicherbereiche. Sie beträgt bei vollem Speicher maximal 5 Minuten.

2.9.6. Testhilfsmittel/Geräteeinstellungen

entfällt

2.9.7. Arbeit unter LPROEM

Aufruf: Eingabe des Namens ZSPE20
Generierung: siehe unter Pkt. 2.9.9. und LPROEM
Fehlerinformation: siehe unter Pkt. 2.9.10.

2.9.8. Arbeit ohne LPROEM

Start: Programmanfangsadresse +14
Generierhalt: Programmanfangsadresse +22 (s. Pkt. 2.9.9.)
Fehlerhalt: Programmanfangsadresse +74 (s. Pkt. 2.9.10.)
Programmende: Programmanfangsadresse +62

Bei auftretenden Unterbrechungen über Vektoren wird auf der jeweiligen Vektoradresse +4 gehalten.

2.9.9. Generierinformationen

Folgende Generiermöglichkeiten sind vorhanden, wobei der Standardfall in Klammern angegeben ist:

7002 PSW : (000000)
Eintragung der gewünschten Priorität für das Prüfprogramm

7004 Teststeuerwort : (100002)
Bit1 = L für Einzeltest

7006 Einzelroutinesteuerwort : (000003)
Bit0 = L und Bit1 = L bedeutet, zwei Routinen sind vorhanden und werden im Einzeltest abgearbeitet.

7010 Dauertestroutinesteuerwort : (000003)
Die vorhandenen 2 Routinen sollen im Dauertest abgearbeitet werden.

7012 Zykluszähler Prüfprogramm : (000001)
000000 ... 177777

7014 Zykluszähler Routine : (000001)
000000 ... 177777

Von den folgenden Generierinformationswerten sind unbedingt welche zu generieren, andernfalls würde der Inhalt beschreibbarer Speicherbereiche geändert. Hier wird kein Standardfall angegeben.

7016 Anfangsadresse Bereich 1

7020 Endadresse Bereich 1

7022 Anfangsadresse Bereich 2

7024 Endadresse Bereich 2

7026 Inhalt = 0, wenn 1 Bereich
Inhalt ist ungleich 0, wenn 2 Bereiche

Die Eintragungen erfolgen bei Arbeitsweise mit LPROEM im Kommandoeingabezustand des LPROEM (s. LPROEM) bzw. bei Arbeitsweise ohne LPROEM im Bedienzustand des Konsolrechners.

2.9.10. Fehlerinterpretation

2.9.10.1. Arbeitsweise mit LPROEM

Die Fehlerinformation, die vom Fehlerausgabegerät angezeigt wird, hat folgendes Format:

SP ZSPE20 RTXX AD SW IS SL PC

SP	Speicherfehler
ZSPE20	Programmname
RTXX	Routinenummer
AD	Fehleradresse
SW	Testart
	Inhalt = 1 : Marching-Test, aufwärts, Wort
	Inhalt = 2 : Marching-Test, aufwärts, gerad. Byte
	Inhalt = 4 : Marching-Test, abwärts, Wort
	Inhalt = 10: Marching-Test, abwärts, ungerad.Byte
	Inhalt = 20: Regeneriertest
IS	Istwert
SL	Sollwert
PC	PC des Fehler-TRAP

Die Bedeutung anderer Fehlerausschriften ist im Pkt. 2.2.9.2. erläutert.

2.9.10.2. Arbeitsweise ohne LPROEM

Im Fehlerfall können den Registern folgende Informationen entnommen werden:

R0	Testart
R1	Fehleradresse
R2	Istwert
R3	Sollwert

Das Verzeichnis der Testarten ist unter Pkt. 2.9.10.1. beschrieben.

2.9.11. Funktionsbeschreibung

In Routine 1 ist der sogenannte Marching-Test realisiert, d.h. durch den gesamten Testbereich läuft vorwärts und rückwärts ein O/L- und L/O-Sprung. Daraus ergeben sich 4 Routineteile und 4 Testdurchläufe des Speicherbereichs. In der Testvorbereitung werden alle Speicherzellen mit der Ausgangsinformation 125252 oktal beschrieben.
Beim Prüfen aufwärts wird erst ein Wort mit der Sollinformation verglichen. Danach wird das gerade Byte mit dem Komplement beschrieben, anschließend gelesen und verglichen. Dann wird das ungerade Byte mit dem Komplement beschrieben. Auf diese Weise wird Wort für Wort behandelt, bis der gesamte Bereich abgearbeitet ist.
Das Prüfen abwärts beginnt mit dem Soll-Istwertvergleich eines Wortes. Anschließend wird das ungerade Byte mit dem Komplement beschrieben und geprüft. Schließlich wird das gerade Byte mit dem Komplement beschrieben.
Routine 2 realisiert den Regeneriertest.
Hierbei wird in aufsteigender Reihenfolge von einer Speicherzelle aus die übernächste, von dieser wiederum die übernächste usw. mit einem Schiebemuster beschrieben. Danach werden die ausgesparten Zellen jeweils mit dem Komplement des Schiebemusters solange beschrieben, bis eine festgelegte Zeit abgelaufen ist. Schließlich werden die zuerst beschriebenen Speicherzellen gelesen und mit den Sollwerten verglichen.
Anschließend werden die ausgesparten Zellen auf die gleiche Art geprüft.
Nach Abschluß dieses Teiles wird der gesamte Test noch einmal mit dem Komplement der Prüfinformation durchgeführt.
Mit diesem Testprogramm kann der beschreibbare Speicher mit Ausnahme des Speicherbereiches des Testprogramms selbst und des zum Testprogramm zugehörigen Arbeits- und Generierbereich im Einzeltest ohne Leitprogrammbenutzung beschrieben werden. Die Größe des Speicherbereichs des Testprogramms ist Pkt. 2.9.4. zu entnehmen. Der Arbeits- und Generierbereich besteht aus den Speicherplätzen 5000 ... 5050, 7000 ... 7030, 7300 und 7400 ... 7430. Der Stackpointer ist auf 2400 gelegt, es wird nur ein Unterprogrammansprung durchgeführt.

2.10. Beschreibung des Prüfprogramms ZTRAPI

2.10.1. Benennung

Test sämtlicher TRAP-Befehle, verbotenen Befehle, T-Bit-TRAP und Stacküberlauf im Konsolrechner

2.10.2. Dokumentationsnummer

Das auf der Diskette DS0 geladene Prüfprogramm ZTRAPI hat die
Dokumentationsnummer 1.57.700208.5 ,
die dazugehörige Programmliste 1.57.700233.3

2.10.3. Programmumfang

Das Programm enthält die folgenden 4 Routinen:

Routinenummer	Routinename	Routinefunktion
RT01	TRAP	Test der Befehle BPT, IOT, EMT und TRAP

RT02	STCK	Test des Stacküberlaufs
RT03	TBIT	Test des T-Bit
RT04	VBEF	Test der verbotenen Befehle

2.10.4. Speicherbedarf

000000...007777

2.10.5. Ausführungszeit

Die Ausführungszeit beträgt ca. 1 Minute.

2.10.6. Testhilfsmittel/Geräteeinstellungen

entfällt

2.10.7. Arbeit unter LPROEM

Aufruf: Eingabe des Namens ZTRAPI
Generierung: siehe unter Pkt. 2.10.9. und LPROEM
Fehlerinformation: siehe unter Pkt. 2.10.10.

2.10.8. Arbeit ohne LPROEM

Start: Programmanfangsadresse +14
Generierhalt: Programmanfangsadresse +22 (s. Pkt. 2.10.9.)
Fehlerhalt: Programmanfangsadresse +74 (s. Pkt. 2.10.10.)
Programmende: Programmanfangsadresse +62

Bei auftretenden Unterbrechungen über Vektoren wird auf der jeweiligen Vektoradresse +4 gehalten.
Die Fortsetzung der Programmabarbeitung vom Generierhalt aus erfolgt durch die Eingabe des Buchstaben P am Bediengerät. Wenn das Programm eingelesen wird, wird es automatisch angesprungen. Dadurch wird sofort der Generierhalt erreicht.

2.10.9. Generierinformationen

Folgende Generiermöglichkeiten sind vorhanden, wobei der Standardfall in Klammern angegeben ist:

7002 PSW : (000000)
Eintragung der gewünschten Priorität für das Prüfprogramm

7004 Teststeuerwort : (100002)
Bit1 = L für Einzeltest

7006 Einzeltestroutinesteuerwort : (000017)
Wert 17 bedeutet alle 4 möglichen Routinen werden im Einzeltest abgearbeitet.

7010 Auswahl der Dauertestroutinen : (000017)
Wert 17 bedeutet alle 4 möglichen Routinen werden im Dauertest abgearbeitet.

7012 Zykluszähler Prüfprogramm : (000001)
000000 ... 177777

7014 Zykluszähler Routine : (000001)
000000 ... 177777

2.10.10. Fehlerinterpretation

2.10.10.1. Arbeitsweise mit LPROEM

In Routine 1 können folgende Fehleranzeigen auftreten:

- IF ZTRAPI RT01 TC TA IS SL PC
  IF Interruptfehler
  ZTRAPI Programmname
  RT01 Routinenummer
  TC TRAP-Code
  TA Testart
  Inhalt = 1 : altes PSW falsch aus Stack geholt
  Inhalt = 2 : Behandlungsprogramm nicht angesprungen
  Inhalt = 4 : SP falsch erniedrigt
  Inhalt =10 : geretteter PC falsch
  Inhalt =20 : altes PSW falsch auf SP gebracht
  Inhalt =40 : neues PSW falsch eingetragen
  IS Istwert
  SL Sollwert
  PC PC des Fehler-TRAP

In Routine 2 können folgende Fehleranzeigen auftreten:

- IF ZTRAPI RT02 400 TA IS SL PC
  Die Bedeutung der Abkürzungen ist die gleiche wie in Routine 1

- IF ZTRAPI RT02 401 TA IS SL PC
  401 Stack wird von 430 erniedrigt
  TA Testart
  Inhalt = 1 : SP wird fehlerhaft erniedrigt
  Inhalt = 2 : Bei 400 erfolgt kein TRAP, in diesem Fall sind IS und SL ohne Bedeutung

- IF ZTRAPI RT02 402 TA IS SL PC
  402 Stack wird unter 400 gesetzt

- IF ZTRAPI RT02 403 TA IS SL PC
  403 Unterprogramm-Ansprung verursacht Stacküberlauf
  TA Testart
  Inhalt = 1 : TRAP wird nicht ausgelöst
  Inhalt = 2 : SP wird fehlerhaft erniedrigt

- IF ZTRAPI RT02 404 TA IS SL PC
  404 IOT-Befehl verursacht Stacküberlauf
  TA hat die gleichen Bedeutungen wie in der vorhergehenden Anzeige.

- IF ZTRAPI RT02 405 TA IS SL PC
  405 JSR-Befehl arbeitet mit Stack falsch
  TA Testart
  Inhalt = 1 : Verbindungs-Register-Inhalt auf Stack falsch
  Inhalt = 2 : Rückkehradresse falsch in Verbindungsregister eingetragen

In Routine 3 können folgende Fehleranzeigen auftreten:

- IF ZTRAPI RT03 20 TA IS SL PC
  Die Bedeutung der Abkürzungen ist wieder die gleiche wie in Routine 1.

- IF ZTRAPI RT03 21 TA IS SL PC
  21 IOT-Befehl bei T-Bit
  TA Testart
  Inhalt = 1 : falscher Rücksprung
  Inhalt = 2 : fehlerhafte Abarbeitung des IOT-Behandlungsprogramm

- IF ZTRAPI RT03 22 TA IS SL PC
  22 Stacküberlauf bei T-Bit
  TA Testart
  Inhalt = 1 : falscher Rücksprung
  Inhalt = 2 : fehlerhafte Abarbeitung des Behandlungsprogramms

- IF ZTRAPI RT03 23 TA IS SL PC
  23 T-Bit-Rücksetzen nach T-Bit-Setzen wirkt nicht
  TA, IS, SL sind hier ohne Bedeutung

- IF ZTRAPI RT03 24 O IS SL PC
  24 T-Bit-TRAP wird 5mal hintereinander ausgeführt

In Routine 4 können folgende Fehlermeldungen auftreten:

- IF ZTRAPI RT04 TC TA IS SL PC
  TC TRAP-Code des verbotenen Befehls
  Die Bedeutung der übrigen Abkürzungen ist wieder die gleiche wie in Routine 1.

- IF ZTRAPI RT04 4304 TA IS SL PC
  4304 JSR-Befehl mit Registermodus
  TA Testart
  Inhalt = 1 : keine TRAP-Ausführung
  Inhalt = 2 : falsche PC-Rücksprungadresse

- IF ZTRAPI RT04 102 TA IS SL PC
  102 JMP-Befehl mit Registermodus
  TA hat die gleichen Bedeutungen wie in der vorhergehenden Anzeige.

Die Bedeutung anderer Fehlerausschriften ist im Pkt. 2.2.9.2. erläutert.

2.10.10.2. Arbeitsweise ohne LPROEM

Fehleranzeigen in Routine 1:

- R0 TRAP-Code des zu testenden TRAP-Befehls
  R1 Testart siehe unter Punkt 2.10.10.1.
  R2 Istwert

R3	Sollwert

Fehleranzeigen in Rouine 2:

- RO	400 bedeutet Stacküberlauffehler
R1	Testart siehe unter Punkt 2.10.10.1. Routine 1
R2	Istwert
R3	Sollwert
- RO	401 Stack wird von 430 erniedrigt
	402 Stack wird unter 400 eingestellt
	403 Unterprogramm-Ansprung verursacht Stacküberlauf
	404 IOT-Befehl verursacht Stacküberlauf
	405 JST-Befehl arbeitet mit Stack falsch
R1	Testart siehe unter Punkt 2.10.10.1. Routine 2
R3	Istwert
R4	Sollwert

Fehleranzeigen in Routine 3:

- RO	20 bedeutet T-Bitfehler
R1	Testart siehe unter Punkt 2.10.10.1. Routine 1
R2	Istwert
R3	Sollwert
- RO	21 IOT-Befehl bei T-Bit
	22 Stacküberlauf bei T-Bit
	23 T-Bit-Rücksetzen nach T-Bit-Setzen wirkt nicht
	24 T-Bit-TRAP wird 5mal hintereinander ausgeführt
R1	Testart siehe unter Punkt 2.10.10.1. Routine 3
R2	Istwert
R3	Sollwert

Fehleranzeigen in Routine 4:

- RO	TRAP-Code des verbotenen Befehls
R1	Testart siehe unter Punkt 2.10.10.1. Routine 1
R2	Istwert
R3	Sollwert
- R0	4304 bedeutet JSR-Befehl mit Registermodus
	102 bedeutet JMP-Befehl mit Registermodus
R1	Testart siehe unter Punkt 2.10.10.1. Routine 4
R2	Istwert
R3	Sollwert

2.10.11. Funktionsbeschreibung

In Routine 1 werden die TRAP-Befehle BPT, IOT, EMT und TRAP getestet. Die Abarbeitung der Befehle erfolgt im RAM-Bereich, so daß die gleiche Befehlsfolge dafür verwendet werden kann, wobei nur der entsprechende Operations-Code immer verändert wird. Die Prüfung beinhaltet den Sprung ins Behandlungsprogramm, das richtige Retten des PSW und des PC, das richtige Erniedrigen des Stack, das Eintragen des neuen PSW sowie das richtige Zurückbringen des alten PSW.

Routine 2 testet den Stacküberlauf-TRAP. Hierbei werden die gleichen Prüfungen wie in Routine 1 vorgenommen. Außerdem wird getestet, ob der Ueberlauf genau bei

Unterschreiten von 400 erfolgt, sowie ob sich der SP auch unter 400 einstellen läßt, ohne daß sofort ein TRAP erfolgt. Weiterhin werden die Stacküberläufe durch Unterprogramm-Benutzung und IOT-Befehl- Abarbeitung erzeugt und geprüft. Schließlich wird getestet, ob beim JSR-Befehl Verbindungsregisterinhalt auf dem Stack gerettet wird und ob die Rückkehradresse ins Verbindungsregister gebracht wird.

Routine 3 befaßt sich mit der Testung des T-Bit-TRAP. Zuerst werden die gleichen Prüfungen wie in Routine 1 ausgeführt. Dann erfolgt die Ausführung von bestimmten Befehlen im T-Bit-Modus, wie IOT-Abarbeitung und Stacküberlaufauslösung. Es wird getestet, ob nach Setzen und anschließendem Rücksetzen des T- Bit ein TRAP ausgeführt wird.
Schließlich werden mehrere hintereinander ausgeführt.

In Routine 4 wird überprüft, ob alle verbotenen Befehle einen TRAP auslösen. Dabei werden die gleichen Prüfungen wie in Routine 1 abgearbeitet.
Weiterhin wird getestet, ob der JMP- und JSR-Befehl im Registermodus einen TRAP verursachen.

2.11. Beschreibung des Prüfprogrammes ZBUSCI

2.11.1. Benennung

Test des Busfehlers ungerade und nicht vorhandene Busadresse (RT01), des ZVE-Taktgebers des K 1620 (RT02) sowie des KBUS zwischen Konsolrechner und K 1840 einschliesslich der 12 ROM's auf dem CIB 8929 (Konsolinterface) (RT03).

2.11.2. Dokumentationsnummer

Das auf der Diskette DS0 geladene Prüfprogramm ZBUSCI hat die
Dokumentationsnummer 1.57.700209.3 ,
die dazugehörige Programmliste 1.57.700234.1 .

2.11.3. Programmumfang

Routine-Nr.	Routinename	Routinefunktion
RT01	BUSF	Busfehlertest
RT02	ZEIT	50 Hz-ZVE-Taktgebertest
RT03	CIBROM	KBUS- und CIBROM-Test

2.11.4. Speicherbedarf

000000 ... 004777

2.11.5. Ausführungszeit

Bei Standardgenerierung <= 10 Sekunden

2.11.6. Testhilfsmittel/Geräteeinstellung

ACHTUNG: Routine 03 erfordert ON-LINE-Betrieb des Konsolrechners (Kopplung zwischen K 1840 und Konsolrechner).

2.11.7. Arbeit unter LPROEM

Aufruf: Eingabe des Namens ZBUSCI
Generierung: siehe unter Punkt 2.11.9. und LPROEM
Fehlerinterpretation: siehe unter Punkt 2.11.10.

2.11.8. Arbeit ohne LPROEM

Start: Programmanfangsadresse + 14
Generierhalt: Programmanfangsadresse + 22 (s. Pkt. 2.11.9.)
Fehlerhalt: Programmanfangsadresse + 74 (s. Pkt. 2.11.10.)
Programmende: Programmanfangsadresse + 62

Bei auftretenden Unterbrechungen über Vektoren wird auf der jeweiligen Vektoradresse + 4 angehalten.
Die Fortsetzung der Programmabarbeitung vom Generierhalt aus erfolgt durch die Eingabe des Buchstaben P am Bediengerät. Wenn das Programm eingelesen wird, wird es automatisch angesprungen.

2.11.9. Generierinformationen

Folgende Generiermöglichkeiten sind vorhanden, wobei der Standardfall in Klammern angegeben ist:

7002 PSW: (000000)
Eintragung der gewünschten Priorität für das Prüfprogramm
7004 Teststeuerwort: (100002)
Bit15=L (konstant für Konsolrechner), Bit1=L für Einzeltest
7006 Einzelroutinesteuerwort: (000007)
Alle drei vorhandenen Routinen werden im Einzeltest abgearbeitet
7010 Dauertestroutinesteuerwort: (000007)
Alle drei vorhandenen Routinen werden im Dauertest abgearbeitet
7012 Zykluszähler Prüfprogramm: (000001)
000000 ... 177777
7014 Zykluszähler Routine: (000001)
000000 ... 177777
7016 Nichtvorhandene Speicheradresse: (000000)
Inhalt 000000 bedeutet Speicherzelle 160000 wird als nichtvorhandene Speicheradresse getestet. Soll eine andere Speicherzelle getestet werden, so ist deren Adresse direkt einzutragen.
7020 Protokollschalter (PROT) für Routine 03: (000000)
PROT < 0: Vollständiges Protokoll nur im Fehlerfall
" = 0: Nur Ausgabe der Kopfzeile (VERSION)
" > 0: Vollständiges Protokoll über alle 12 ROM's auf dem CIB (CIBROM's)

ANMERKUNG: Unabhängig vom Inhalt von PROT erscheint im Fehlerfall immer ein vollständiges Protokoll.

7022 Sollprüfsumme CIBROM 0001: (016477 = 13DFH)

	ROM für 1. Tetrade im Bereich 140000 - 143777
7024	Sollprüfsumme CIBROM 0002: (017141 = 1E61H)
	ROM für 2. Tetrade im Bereich 140000 - 143777
7026	Sollprüfsumme CIBROM 0003: (014171 = 1879H)
	ROM für 3. Tetrade im Bereich 140000 - 143777
7030	Sollprüfsumme CIBROM 0004: (010277 = 10BFH)
	ROM für 4. Tetrade im Bereich 140000 - 143777
7032	Sollprüfsumme CIBROM 0005: (014041 = 1821H)
	ROM für 1. Tetrade im Bereich 144000 - 147777
7034	Sollprüfsumme CIBROM 0006: (015711 = 1BC9H)
	ROM für 2. Tetrade im Bereich 144000 - 147777
7036	Sollprüfsumme CIBROM 0007: (012105 = 1445H)
	ROM für 3. Tetrade im Bereich 144000 - 147777
7040	Sollprüfsumme CIBROM 0008: (006567 = 0D77H)
	ROM für 4. Tetrade im Bereich 144000 - 147777
7042	Sollprüfsumme CIBROM 0009: (013423 = 1713H)
	ROM für 1. Tetrade im Bereich 150000 - 153777
7044	Sollprüfsumme CIBROM 0010: (015771 = 1BF9H)
	ROM für 2. Tetrade im Bereich 150000 - 153777
7046	Sollprüfsumme CIBROM 0011: (012662 = 15B2H)
	ROM für 3. Tetrade im Bereich 150000 - 153777
7050	Sollprüfsumme CIBROM 0012: (006062 = 0632H)
	ROM für 4. Tetrade im Bereich 150000 - 153777

Die Eintragungen erfolgen bei Arbeitsweise mit LPROEM im Kommandoeingabezustand des LPROEM (s. LPROEM) bzw. bei Arbeitsweise ohne LPROEM im Bedienzustand des Konsolrechners.

2.11.10. Fehlerinterpretation

2.11.10.1. Arbeitsweise mit LPROEM

In Routine 01 kann folgende Fehlermeldung auftreten:

IF ZBUSCI RT01 AD SW IS SL PC IV

IF	Interruptfehler
ZBUSCI	Programmname
RT01	Routinenummer
AD	Adresse
	Inhalt = 1421: ungerade Adresse
	Inhalt = 160000 oder generierter Wert: nichtvorhandene Adresse
SW	Steuerwort
	Inhalt = 1 : altes PSW falsch aus Stack geholt
	Inhalt = 2 : Behandlungsprogramm nicht angesprungen
	Inhalt = 4 : SP falsch erniedrigt
	Inhalt = 10 : geretteter PC falsch
	Inhalt = 20 : altes PSW falsch auf SP gebracht
	Inhalt = 40 : neues PSW falsch eingetragen
IS	Istwert
SL	Sollwert
PC	PC des Fehlertrap
IV	Interruptvektor

Bei der Abarbeitung von Routine 02 kann folgende Fehlermeldung auftreten:

ZF ZBUSCI RT02 AD SW IS SL PC IV

```
  ZF  Zeitfehler
  AD  Adresse
      Inhalt = 177546 : Geräteadresse des Uhrtaktgebers
  SW  Steuerwort
      Inhalt = 1 : Test ohne Interrupt
      Inhalt = 2 : Test mit Interrupt
  Alle übrigen Bezeichnungen entsprechen denen vom Fehler in Routine 01.
```

In Routine 03 entsteht im Fehlerfall bedingungslos folgende Fehlerausschrift:

TEST DER BAA UND DER CIBROMS: DD-MMM-YY*VERSION*(146733)-(001355) ??

```
CIBROM-Nr.  SOLL:   IST:

   0001     SSSS   IIII  <------- FEHLER
   0002     "      "
   0003     "      "
   0004     "      "
   0005     "      "
   0006     "      "
   0007     "      "
   0008     "      "
   0009     "      "
   0010     "      "
   0011     "      "
   0012     SSSS   IIII
```

DF ZBUSCI RT03 PC

DD-MMM-YY	Erstellungsdatum Prüfprogramm
(146733)	Niederwertiger Teil der über alle 12 CIB-ROM's gebildeten Gesamtprüfsumme
(001355)	Höherwertiger Teil der über alle 12 CIB-ROM's gebildeten Gesamtprüfsumme
??	Es wurde ein Fehler zwischen berechneter und auf den CIB-ROM's stehender Gesamtprüfsumme ermittelt.
CIBROM-Nr.	Bedeutung wie in 2.11.9. erläutert.
SSSS	Sollprüfsumme des jeweiligen CIBROM (generierbar)
IIII	Istprüfsumme des jeweiligen CIBROM (berechnet)
<---- FEHLER	zeigt auf einen defekten CIBROM

Nach Ausgabe der Klartext-Fehlermeldung erfolgt eine LPROEM- Fehlermeldung entsprechend Form 7 aus Pkt. 2.2.9.3.:

DF	Datenfehler
ZBUSCI	Programmname
RT03	Routinenummer
PC	PC des Fehlertrap

Die Bedeutung anderer Fehlerausschriften ist im Pkt. 2.2.9.2. erläutert.

2.11.10.2. Arbeitsweise ohne LPROEM

Fehleranzeige in Routine 01:

R0 Adresse
1421 bedeutet ungerade Adresse
160000 oder generierter Wert bedeutet nichtvorhandene Speicheradresse
R1 Steuerwort
Inhalt siehe unter Punkt 2.11.10.1. Routine 01
R2 Istwert
R3 Sollwert

Fehleranzeige in Routine 02:

R0 Adresse
177546 bedeutet Geräteadresse des Uhrtaktgebers
R1 Steuerwort
1 bedeutet Test ohne Interrupt
2 bedeutet Test mit Interrupt
R2 Istwert
R3 Sollwert

Fehleranzeige in Routine 03:

Es erscheint diesselbe Klartext-Fehlermeldung wie unter Pkt. 2.11.10.1. beschrieben.
Aus folgenden Zellen kann folgende Information entnommen werden:

5100	SUMIST	Berechnete Einzelprüfsumme CIBROM 0001
.		.
.		.
.		.
5126		Berechnete Einzelprüfsumme CIBROM 0012
5200	SUMLOW	Berechnete Gesamtprüfsumme niederw. Teil
5202	SUMHIG	Berechnete Gesamtprüfsumme höherw. Teil
7022	SMSOLL	Soll-Einzelprüfsumme CIBROM 0001
.	.	.
.	.	.
.	.	.
7050		Soll-Einzelprüfsumme CIBROM 0012

2.11.11. Funktionsbeschreibung

In Routine 01 werden die beiden verschiedenen Arten des Busfehlers getestet. Busfehler ungerade Adresse wird erzeugt, indem versucht wird, die Speicheradresse 1421 indirekt zu besprechen.
Busfehler nichtvorhandene Busadresse wird erzeugt, indem versucht wird, je nach Generierung die Busadresse 160000 oder irgendeine andere nichtvorhandene anzusprechen. Dabei wird überprüft, ob nach Busfehlertrap das alte PSW wieder eingetragen wird, ob das Busfehler-Behandlungsprogramm angesprungen wird, ob die richtige PC-Rücksprungadresse im SP abgespeichert wird, ob das alte PSW richtig im Stack gerettet wird und ob das neue PSW richtig eingetragen wurde.

In Routine 02 wird die richtige Arbeitsweise des ZVE-Taktgebers geprüft. Es wird einmal die Arbeitsweise ohne Interrupt und einmal mit Interrupt kontrolliert. Dabei wird eine bestimmte Anzahl von Impulsen aufgezählt.

In Routine 03 werden der KBUS (Bus zwischen KR und K 1840) und die 12 ROM's auf dem CIB 8929 geprüft. Zuerst wird eine Gesamtprüfsumme über den Adreßbereich 140000 ... 153772 gebildet, indem wortweise die Inhalte dieser Zellen in SUMLOW aufaddiert und die dabei entstehenden Überträge in SUMHIG registriert werden. Es wird kontrolliert, ob der Inhalt in SUMLOW (niederwertiger Teil der Gesamtprüfsumme) mit dem auf ROM- Adresse 153774 stehenden Inhalt übereinstimmt und ob der Inhalt in SUMHIG (höherwertiger Teil der Gesamtprüfsumme) mit dem auf ROM-Adresse 153776 stehenden Inhalt übereinstimmt. Anschließend werden 12 Einzelprüfsummen berechnet, für jeden CIB-ROM eine. Dabei wird die Prüfsumme durch Aufsummieren der logischen Werte modulo 16 gebildet.
Die berechneten Einzelprüfsummen werden fortlaufend wortweise in den Zellen SUMIST abgespeichert. Es wird kontrolliert, ob die vom Prüfprogramm berechneten Istprüfsummen mit den berechneten Sollprüfsummen übereinstimmen. Diese stehen fortlaufend in SMSOLL.

2.12. Beschreibung des Prüfprogramms AKSAIS

2.12.1. Benennung

Test der AIS (Anschlußsteuerung für Interface, seriell) für IFSS und S2 mit Kurzschlußsteckern

2.12.2. Dokumentationsnummer

Das auf der Diskette DS0 geladene Prüfprogramm AKSAIS hat die
Dokumentationsnummer 1.57.700210.8 ,
die dazugehörige Programmliste 1.57.700235.8

2.12.3. Programmumfang

Das Prüfprogramm AKSAIS besteht aus nachfolgend aufgeführten Einzelroutinen:

Routine-Nummer	Routine-Name	Routine-Quittung	Routine-Funktion
RT01	ZKO	0	Test des Zeitkanals 0
RT02	ZK1	1	Test des Zeitkanals 1
RT03	ZK2	2	Test des Zeitkanals 2
RT04	ZK3	3	Test des Zeitkanals 3
RT05	IPRF	I	Test der Interruptprioritätsfolge
RT06	-		Nicht belegt
RT07	DK1A	A	Test des Datenkanals 1A
RT10	DK1B	B	Test des Datenkanals 1B
RT11	DK2A	C	Test des Datenkanals 2A
RT12	DK2B	D	Test des Datenkanals 2B
RT13	BREAK	H	Test der Break-Halt-Erzeugung und der Bedienkanalumschaltung
RT14	-		Nicht belegt
RT15	SS1A	a	Test der Statussteuerung und des Synchronbetriebes in 1A
RT16	SS1B	b	Test der Statussteuerung und des Synchronbetriebes in 1B
RT17	SS2A	c	Test der Statussteuerung und des Synchronbetriebes in 2A

RT20	SS2B	d	Test der Statussteuerung und des Synchronbetriebes in 2B

RT13 nur bei IFSS aber nicht bei 1. AIS möglich.
RT15 bis 20 nur bei S2 möglich.

2.12.4. Speicherbedarf

000000 ... 014777

2.12.5. Ausführungszeit

Die Laufzeit für einen Programmzyklus aller Routinen dauert maximal etwa eine Minute. Sie ist von der Routinegenerierung des Prüfprogramms abhängig. Die Routinen RT01 ... RT04 müssen jeweils 5 Sekunden laufen, dann funktioniert die Zeitbasis der Zeitkanäle richtig.
Für die Routine RT13 kann keine Laufzeit angegeben werden, da die Break-Halt-Erzeugung in den Rechner-Bedienzustand führt und quittiert werden muß.

2.12.6. Testhilfsmittel/Geräteeinstellungen

Als Testhilfsmittel werden 1 bis 4 IFSS-Kurzschlußstecker 051-2000 und ein bis zwei S2-Kurzschlußstecker 051-2290 benötigt, je nach zu testender IFSS- bzw. S2-Kanäle pro AIS-Steckeinheit (STE). Mögliche, mit diesem Prüfprogramm zu testende AIS-Steckeinheitentypen sind:

STE 051-1750 (4 x IFSS)
STE 051-1760 (2 x IFSS, 1 x S2)
STE 051-1761 (2 x IFSS)
STE 051-1770 (2 x S2)

Als Geräteeinstellungen können die Wickelstiftbelegungen auf der zu testenden AIS-Steckeinheit angesehen werden. Da das Prüfprogramm für 4 Steckeinheiten-Standards verwendet werden soll, muß es jedesmal entsprechend generiert werden (s. Pkt. 2.12.9.). Die Bedeutung der Wickelstiftbelegung muß der Einstellvorschrift der technischen Beschreibung der AIS entnommen werden.

2.12.7. Arbeit unter Leitprogramm

Aufruf:	Eingabe des Namens AKSAIS (s. LPROEM)
Generierung:	s. Pkt. 2.12.9. und LPROEM
Fehlerinterpretation:	s. Pkt. 2.12.10.

Für die Prüfung der STE 051-1750 liegt die Generierung als Standardgenerierung vor.

2.12.8. Arbeit ohne Leitprogramm

Prüfprogrammstart	: Ladeadresse +14
Generierungshalt	: Ladeadresse +22
Programmendehalt	: Ladeadresse +62
Fehlerhalt	: Ladeadresse +74

Treten Interrupts auf, die durch das Prüfprogramm nicht ausgelöst werden, geht der Rechner auf der entsprechenden Vektoradresse +4 in den Haltzustand.

2.12.9. Generierinformationen

In die Adressen ab 7002 können beliebige, sinnvolle Generierinformationen als Oktalwörter eingetragen werden.

```
7002 : Prozessorstatuswort (PS) mit Prioritätsebene so hoch,
       daß zu testende AIS-Interrupts wirksam werden können.
       Falls Bit 0 = L : Ausgabe der Routinequittung auf Bildschirm.
7004 : Teststeuerwort, Bit 1 = L; Einzeltest
7006 : Einzeltestroutinesteuerwort (RS), Routinezuordnung
       entsprechend der Reihenfolge unter Pkt. 2.12.3.
       RT01 = Bit 0 ... RT20 = Bit 15
7010 : Dauertestroutinesteuerwort, Generierung nur durch LPROEM
7012 : Zykluszähler Prüfprogramm (PZ), 0 ... 177777
7014 : Zykluszähler Routine (RZ), 0 ... 177777

Prüfprogrammspezifische Generierung (GN)

7016 : AIS-STE-Typ und -Modifizierung
       Bit 0 = 0/L ; Bedienung an 1A      : nein/ja
                     (ja bewirkt: Kennung = 177570,
                                  Adresse = 177500 und
                     Unterdrückung der Routinen 1, 7, 13)
       Bit 1 = 0/L ; Gerät an 1B          : nein/ja
       Bit 2 = 0/L ; Gerät an 2A          : nein/ja
       Bit 3 = 0/L ; Gerät an 2B          : nein/ja
       Bit 4 = 0/L ; Bedienkanalum-
                     schaltung verboten   : nein/ja
       Bit 5 = 0/L ; Breaklänge           : 10 Bit/9 Bit
       Bit 6 = 0/L ; Kanal 1B (S2)        : Datenkanal/Hilfskanal
       Bit 7 = 0/L ; Kanal 2B (S2)        : Datenkanal/Hilfskanal
       Bit 8   nicht belegt
       Bit 9 = 0/L ; 2. U856 fehlt        : nein/ja (bei STE 7161)
       Bit 10 und 11 nicht belegt
       Bit 12, 13, 14 = L : STE-Typ       : 1770      (7)
       Bit 13, 14 = L, Bit 12=0 : STE-Typ: 1760/1761 (6)
       Bit 12, 14 = L, Bit 13=0 : STE-Typ: 1750      (5)
       Bit 15 = 0/L ; ZK-Zuordnung von Bedien/ oder Standard-AIS
                      (unabhängig vom STE-Typ) abweichend
                                          : nein/ja
7020 : AIS-Auswahl 4 aus 16
7022 : AIS-Vergleichsadresse, unmodifiziert
7024 : Zeitkanal mit Vorteiler(P) und Zeitkonstante(TC) für 1A
       im Asynchronbetrieb
       Bit 0 bis 7   : Zeitkonstante TC = 1 ... 256,0
       Bit 8 = 0/L   : Vorteiler P = 0/L : 16/256
       Bit 9 und 10  : Zeitkanal 0 ... 3
7026 : Zeitkanal mit P und TC für 1B im Asynchronbetrieb
       Bitbelegung wie oben
7030 : Zeitkanal mit P und TC für 2A im Asynchronbetrieb
       Bitbelegung wie oben
7032 : Zeitkanal mit P und TC für 2B im Asynchronbetrieb
       Bitbelegung wie oben
7034 : Zeitkanal mit P und TC für 1A im Synchronbetrieb
```

Bitbelegung wie oben
7036 : Zeitkanal mit P und TC für 1B im Synchronbetrieb
Bitbelegung wie oben
7040 : Zeitkanal mit P und TC für 2A im Synchronbetrieb
Bitbelegung wie oben
7042 : Zeitkanal mit P und TC für 2B im Synchronbetrieb
Bitbelegung wie oben
7044 : K 1600-Interruptanfangsvektor

Standardgenerierung

Generieradresse	Generierinformation	LPRO-Kennzeichen
7002	000140	PS
7004	000002	
7006	173737	RS
7010	171737	
7012	000001	PZ
7014	000001	RZ
7016	050000	GN
7020	177564	GN
7022	174000	GN
7024	000001	GN
7026	001001	GN
7030	002001	GN
7032	003001	GN
7034	000004	GN
7036	001004	GN
7040	002004	GN
7042	003004	GN
7044	000300	GN

Die Eintragung auf der Adresse 7016 hat bei den STE-Typen 1760/1770 die automatische Umgenerierung der Zeitkanalzuordnung (zum Standardfall für 1760/1770) und bei den STE-Typen 1750/1760/1761 im Bedien-AIS-Fall die automatische Umgenerierung der AIS-Auswahl zu 177570 und der AIS-Vergleichsadresse zu 177500 zur Folge.
Deshalb ist mit einer Generierinformationsänderung auf der Adresse 7016 direkt oder mit dem Kennzeichen GN über Leitprogramm die Generierung zur Prüfung für folgende AIS-STE-Typen zu erreichen:

AIS-STE Typ	Generierinformation für Adresse 7016	
	Standard	Bedienung
1750	50000	50001
1760	60000	60001
1761	61000	61001
1770	70000	

Befinden sich mehrere AIS-STE im Rechner, muß jede eine andere AIS-Auswahl und eine andere AIS-Vergleichsadresse erhalten.

Die Generierparameter für die AIS des Konsolrechners sind im Abschnitt 2.14.3. enthalten.

2.12.10. Fehlerinterpretation

2.12.10.1. Arbeitsweise mit LPROEM

Folgende Fehler werden durch das Prüfprogramm AKSAIS erkannt:

- ZF AKSAIS RTXX AD SW IS SL PC
  Zeitfehler; der erwartete Interrupt ist ausgeblieben.
  XX = 01 ... 04 : SW = 6
  XX = 05 : SW = 1, 3, 5
  XX = 07 ... 12 : SW = 0,4
  XX = 15 ... 20 : SW = 2
  IS = SL = O
  In SW wird spezifiziert, um welchen ausgebliebenen Interrupt es sich handelt:
  SW = 0 : Sender-Interrupt ausgeblieben
  = 1 : 1. U856-Interrupt ausgeblieben
  = 2 : Status-Interrupt ausgeblieben, R2 = SW von Statusfehler SF
  = 3 : 2. U856-Interrupt ausgeblieben
  = 4 : Empfänger-Interrupt ausgeblieben
  = 5 : U857-Interrupt ausgeblieben
  = 6 : Zeitkanal-Interrupt ausgeblieben

- MF AKSAIS RTXX AD SW IS SL PC
  Maskenfehler; Zeitkanalinterrupt trotz Maskenbit = 0 gekommen.
  XX = 01 ... 04
  SW = IS = SL = O

- SF AKSAIS RTXX AD SW C1 C2 C3 C4
  C5 C6 C7 C8 PC IV
  Statusfehler; Statusregister des U856 (LR0, LR1, LR2/B sind die Ursache.
  XX = 07 ... 12 : SW = 0 ... 7, 11 ... 14
  XX = 15 ... 20 : SW = 0, 21 ... 26
  C5 = C6 = 0
  In SW wird spezifiziert, um welchen Statusfehler es sich handelt:
  SW = 0 : Grundstatus fehlerhaft
  = 1 : Sendepuffer nicht leer
  = 2 : Empfangszeichen nicht verfügbar
  = 3 : Nicht alles gesendet
  = 4 : Interrupt-Bit im Abfragebetrieb gesetzt
  = 5 : Paritätsfehler-Bit gesetzt
  = 6 : Empfangsüberlauf-Bit gesetzt
  = 7 : Interruptvektor nicht konstant geblieben
  = 11: Empfängerüberlauf-Bit nicht gesetzt
  = 12: Paritätsfehler-Bit nicht gesetzt
  = 13: Rahmenfehler-Bit nicht gesetzt
  = 14: Breakfehler-Bit nicht gesetzt
  = 21: DCD-Bit nicht gesetzt (Datenkanal S2)
  = 22: DCD-Bit nicht rückgesetzt (Datenkanal S2)
  = 23: CTS-Bit nicht gesetzt (Datenkanal S2)
  = 24: CTS-Bit nicht rückgesetzt (Datenkanal S2)
  = 25: CTS-Bit nicht gesetzt (Hilfskanal S2)
  = 26: CTS-Bit nicht rückgesetzt (Hilfskanal S2)
  = 31: Status während Blockübertragung im Monobytesynchronbetrieb (MBS)
  = 32: Status während Blockübertragung im Bibytesynchronbetrieb (BBS)
  = 41: Status nach Blockübertragung im MBS
  = 42: Status nach Blockübertragung im BBS
  = 51: Status spezielle Empfangsbedingungen während Blockübertragung

im MBS
= 52: Status spezielle Empfangsbedingungen während Blocküberträgung
im BBS

- DF AKSAIS RTXX AD SW C1 C2 C3 C4
C5 C6 C7 C8 PC IV
Datenfehler; Istwert (C5 und C7) und Sollwert (C6 und C8)
stimmen nicht überein.
XX = 07 ... 12 : SW = 0
XX = 14 ... 20 : SW = 1, 2
In SW wird spezifiziert, um welchen Datenfehler es sich handelt:
SW = 0 : Fehlerhaftes Byte im Asynchronbetrieb
C7 = Sollbyte, C8 = Istbyte, C5 = C6 = 0
SW = 1 : Fehlerhafte Bytesumme des Datenblockes im Synchronbetrieb
C7 = Sollbytesumme, C8 = Istbytesumme, C5 = C6 = 0
SW = 2 : Fehlerhafte CRC-Zeichen
C5, C7 = Ist-CRC-Zeichen
C6, C7 = Soll-CRC-Zeichen

- IF AKSAIS RTXX AD SW C1 C2 C3 C4
C5 C6 C7 C8 PC IV
Interruptfehler; Istvektor (IV) und Sollvektor (C6) stimmen nicht überein
oder der Vektor in LR2 (B) wurde anders als erwartet
modifiziert.
XX = 05, 07 ... 12
C5 = 0
In SW wird spezifiziert, um welchen Interruptfehler es sich handelt:
SW = 1 : Vektor in LR2/B wurde anders als erwartet modifiziert.
Erwartet wurde " Empfangszeichen verfügbar - Modifizierung ".
= 2 : Vektor in LR2/B wurde anders als erwartet modifiziert.
Erwartet wurde " Sendepuffer leer - Modifizierung ".
= 3 : Vektor in LR2/B wurde anders als erwartet modifiziert.
= 4 : Vektorfehler, Interrupt ist über einen anderen Vektor als den
erwarteten gekommen.

Die Fehlerausschriften haben folgende Bedeutung:

XX Routine-Nummer
AD Adresse des den Fehler verursachenden Zeit- oder Datenkanals oder der Sammelmaske
SW Statuswort zur Spezifizierung des jeweiligen Fehlers
IS Istwert
SL Sollwert
PC Befehlszähler zum Fehlerzeitpunkt
C1 Initialisierart, siehe unten
C2 U856-LPO-Inhalt
C3 U856-LR1-Inhalt
C4 U856-LR2/B-Inhalt
C5 Null oder CRC-Istwert (1. Zeichen)
C6 Sollvektor für zu erwartenden Interrupt oder CRC-Istwert (2. Zeichen)
C7 Istwert, Daten
C8 Sollwert, Daten
IV Istvektor, über den der Interrupt gekommen ist

zu C1:
In C1 wird angegeben, um welche Initialisierart des U856-Schaltkreises es sich jeweils handelt. Da dem Schaltkreis dafür 8 Schreibregister SRO ... SR7 zur Verfügung stehen, werden diese 8 Informationen mit einem Kennzeichen in C1

zusammengefaßt. Die Bedeutung der Informationen für die SR0 ... SR7 sind der technischen Beschreibung der AIS K 8060 zu entnehmen. Das Kennzeichen C1 ist so aufgebaut, daß sich in Bit 0 ... 2 die Initialisierart befindet und in Bit 3 die Angabe darüber, ob es sich um Abfragebetrieb (Bit 3 = 0) oder um Interruptbetrieb (Bit 3 = L) handelt. C1 = 6 oder 7 bei Routine 13, C1 = 20 bei Routine 05 und C1 = 31, 32, 41, 42, 51 oder 52 bei Routinen 15 ... 20.

	SR0	SR1	SR2	SR3	SR4	SR5	SR6	SR7	
C1 = 1 :	030	000	000	301	104	150	000	000	8 Bit
= 2 :	030	000	000	101	111	050	000	000	7 Bit
= 3 :	030	000	000	201	117	110	000	000	6 Bit
= 4 :	030	000	000	000	117	010	000	000	5 Bit
= 5 :	030	005	000	000	104	000	000	000	5 Bit
= 6 :	030	000	000	301	104	150	000	000	8 Bit
= 7 :	030	000	000	301	107	150	000	000	8 Bit
=20 :	030	002	000	300	104	150	000	000	8 Bit
=31 :	030	037	000	333	000	357	026	026	8 Bit
=32 :	030	037	000	333	020	353	070	070	8 Bit

SR1 und SR2 werden bei Interruptbetrieb noch entsprechend verändert.

2.12.10.2. Arbeitsweise ohne LPROEM

Bei jedem Fehler geht der Rechner in den Bedienzustand auf der Ladeadresse +74 (s. Pkt. 2.12.8.).
In diesem Zustand können die unter dem Pkt. 2.12.10.1. beschriebenen Fehlerinformationen einzeln aus Registern und Speicherzellen entnommen werden.

XX = 7400	AD = R0	C1 = 7300	C5 = 7310
IV = 7402	SW = R1	C2 = 7302	C6 = 7312
PZ = 7406	IS = R2	C3 = 7304	C7 = 7314
RZ = 7410	SL = R3	C4 = 7306	C8 = 7316
	PC = R4		

Nach der Fehlerauswertung kann mit Taste "P" über das Bediengerät der Testlauf fortgesetzt werden.

2.12.11. Funktionsbeschreibung

Das Prüfprogramm testet im Asynchronbetrieb die serielle Anschlußsteuerung AIS. Sie kann auf 4 verschiedenen Steckeinheiten realisiert sein und die Interfaces IFSS und S2 erzeugen. Es stehen immer 4 Daten-E/A-Kanäle zur Verfügung. Für jeden zu testenden Interfaceanschluß wird ein Kurzschlußstecker benötigt. Jedem zu testenden Datenkanal muß mittels Generierung einer der drei Zeitkanäle zugeordnet werden.
Ein IFSS-Anschluß belegt einen Datenkanal (DK), ein S2-Anschluß belegt zwei Datenkanäle, wobei der B-Kanal auch Hilfskanal sein kann.
Es werden folgende Tests durchgeführt:

- Jeder Zeitkanal im Interruptbetrieb (IFSS oder S2).
- Prioritätsreihenfolge zwischen 1. U856, 2. U856 und U857 (IFSS oder S2).
- Jeder E/A-Kanal im Abfrage- und Interruptbetrieb bezüglich der Datensignale (IFSS oder S2).
- Breakerkennung (9 oder 10 Bit) und falls erlaubt, Umschaltung der Bedienkanäle 1A und 1B (nur IFSS).

- Jeder E/A-Kanal im Interruptbetrieb bezüglich der Statuserzeugung (CTS, DCD) mit DTR und RTS (nur S2).

Pro Zeitkanal wird im Interruptbetrieb bei fehlerfreiem Lauf eine Meßzeit von fünf Sekunden realisiert. Dabei wird jeder Zeitkanal als Zeitgeber mit einer Zeitkonstante von 255 initialisiert. Die über den Interruptvektor auftretenden Interrupts werden gezählt. Anschließend an diesen Test wird die Sperrwirkung der AIS-Maske in jedem Zeitkanal getestet.
Um die Prioritätsreihenfolge der Interrupts zu testen, wird im 1. U856, im 2. U856 (falls vorhanden) und im U857 ein Interrupt ausgelöst und die prioritätsgerechte Abarbeitung der Interrupts getestet.
Pro Datenkanal werden 4 verschiedene Initialisierarten (8 Bit-, 7 Bit-, 6 Bit-, 5 Bit-Übertragung) des U856 im Abfragebetrieb getestet. Dabei werden außerdem Paritätsfehler, Rahmenfehler, Breakfehler und Empfängerüberlauf provoziert.
Anschließend wird wieder mit den 4 Initialisierarten der Interruptbetrieb getestet, wobei die Prioritätsreihenfolge zwischen Sender und Empfänger mit getestet wird.
Jedem Datenkanal kann durch Generierung des Prüfprogramms einer der 3 Zeitkanäle zugeordnet werden.
Die Breakerkennung in den IFSS-Kanälen (1A, 1B) kann je nach Programmgenerierung auf 9 oder 10 Bit ansprechen. Bei richtiger Funktionsweise geht der Rechner bei diesem Test auf der Adresse 5100 oder 5102 in den Bedienzustand (Kanal 1A), mit Taste "P" kann fortgesetzt werden und ein erneuter Bedienzustand auf der Adresse 5104 oder 5106 (Kanal 1B) wird erreicht, der wieder mit Taste "P" quittiert werden muß. Dieses Bild ergibt sich nur, wenn keine Bedienkanalumschaltung vorliegt.
Ist Bedienkanalumschaltung auf der AIS-STE eingestellt und im Prüfprogramm generiert, müssen sich nacheinander folgende Bedienzustände auf den Adressen
1. 5100 oder 5102 ("P")
2. 5110 oder 5112 ("P")
3. 5100 oder 5102 ("P") ergeben.
Die Statussteuerung funktioniert nur in den S2-Kanälen und in den B-Kanälen in Abhängigkeit davon, ob es sich um einen S2-Datenkanal (DK) oder um einen S2-Hilfskanal (HK) handelt (kann durch Wickelbrücke auf AIS-STE und Programmgenerierung gewählt werden).
Der Test selbst wird im Interruptbetrieb durchgeführt und provoziert die Interruptauslösung (CTS- und DCD-Bit) mit den Steuersignalen DTR und RTS in Abhängigkeit von DK oder HK.
Der duplexe Synchronbetrieb ist ebenfalls nur in den S2-Kanälen möglich. Es werden im MBS und BBS Bytekombinationen von 100 ... 377 gesendet, empfangen und sender- und empfangsseitig zu einer Prüfsumme verrechnet. Die gesendeten Zeichen werden ab Speicherbereich 14000 und die empfangenen Zeichen ab Speicherbereich 15000 abgespeichert. Der Prüfsummenvergleich und der CRC-Zeichenvergleich ermöglichen die Bewertung des Synchrontests. Der CRC-Zeichenvergleich wird im MBS mit dem Polynom

$$x^{16}+x^{12}+x^{5}+1$$

und im BBS mit dem Polynom

$$x^{16}+x^{15}+x^{2}+1$$

durchgeführt.

2.13. Beschreibung des Prüfprogrammes GFLOPM

2.13.1. Benennung

Test der Folienspeichereinheit (Anschlußsteuerung für Folienspeicher(AFS) und Laufwerk).

2.13.2. Dokumentationsnummer

Das auf der Diskette DS0 geladene Prüfprogramm GFLOPM hat die
Dokumentationsnummer 1.57.700211.6 ,
die dazugehörige Programmliste 1.57.700236.6 .

2.13.3. Programmumfang

Das Prüfprogramm besteht aus Einzelroutinen:

Routine-Nr.	Funktion
RT01	AFS-Puffer-Test
RT02	Laufwerk-Test
RT03	Formatierung von Disketten
RT04	Setze einfache Dichte
RT05	Setze doppelte Dichte
RT06	Doppeln vom Laufwerk 0 zum Laufwerk 1
RT07	Vergleichen der Disketten beider Laufwerke

2.13.4. Speicherbedarf

0 ... 25777

2.13.5. Ausführungszeit

bis 120 min je nach Generierung und Laufwerktyp

2.13.6. Testgegenstand

Das vorliegende Prüfprogramm GFLOPM ist in der Lage, neben der AFS unterschiedliche, entsprechend angeschlossene Laufwerke zu prüfen.

1 Einschub (jeweils bestückt mit 1 oder 2 Laufwerken):
- Laufwerk MF 6400 (8")
- K 5665 mit 1 oder 2 Laufwerken MF 3200 (8")
- K 5661.10 mit 1 oder 2 Laufwerken K 5600.10 (5 1/4" 40 Spuren)
- K 5661.20 mit 1 oder 2 Laufwerken K 5600.20 (5 1/4" 80 Spuren)
  oder mit 1 oder 2 Laufwerken K 5600.16 (5 1/4" 80 Spuren)
- K 5666.10 mit 1 oder 2 Laufwerken K 5602.10 (8")

Mit dem Prüfprogramm GFLOPM sind auch die dem angeschlossenen Laufwerk entsprechenden Disketten auf ihre Beschreib- und Lesbarkeit zu prüfen. Weiterhin bietet dieses Programm die Möglichkeit der Formatierung von Disketten.

2.13.7. Arbeit unter Leitprogramm LPROFM

Aufruf: Eingabe des Namens GFLOPM
Generierung: siehe 2.13.9.
Fehlerausgabe: siehe 2.13.10.

2.13.8. Arbeit ohne Leitprogramm LPROEM

Start: Programmanfangsadresse +14
Generierhalt: Programmanfangsadresse +22
Programmende: Programmanfangsadresse +62
Fehlerhalt: Programmanfangsadresse +74

Bei Interrupts, deren Ursache nicht in der Folienspeichereinheit zu suchen sind, geht das Programm auf der jeweiligen Vektoradresse +4 in den Halt-Zustand.

2.13.9. Generierinformationen

Die nachfolgend aufgeführten Informationen sind als Oktalworte generierbar und gestatten eine sinnvolle Beeinflussung des Programmablaufes.

Speicher-platz	Bedeutung	(Standard-wert)
7002	Programmstatuswort mit Prioritaetsebene (Priorität muß kleiner/gleich 4 sein)	(000000)
7006	Einzeltestroutinesteuerwert RS	(000003)

BIT	RS	Bedeutung bei BITXX=1
00	1	AFS-Puffer-Test
01	2	LW-Test
02	4	Formatieren
03	10	Setze einfache Dichte
04	20	Setze doppelte Dichte
05	40	Doppeln von LW0 nach LW1
06	100	Vergleichen zwischen LW0 und LW1

7010	Dauertestroutinesteuerwort (Nur die Routinen 1 und 2 können ausgewählt werden)	(000003)
7012	Zykluszähler Prüfprogramm	(000001)
7014	Zykluszähler Routine RZ	(000001)
7016	Testauswahl	(000000)

BIT	Oktal	Bedeutung bei BITXX=1
15	100000	LW1 gewählt
14	040000	LW0 gewählt
13	020000	8"-MFM oder 5 1/4" MFM
12	010000	16 Sektoren (Bit 11 auch gesetzt)
11	004000	5 1/4"-MFM 16 Sektoren oder 26 Sektoren
10	002000	dopp. Dichte (nur bei 8"DEC-SPEZ.: nicht bei 8" MFM oder 5 1/4" MFM.)
09	001000	gelöschte Daten

08
07 000X00 Datenmuster
06

Mit X=0 Zufallsmuster
1 Nur Nullen
2 Nur Einsen
3 Gleitende Null durch Einsen
4 Gleitende Eins durch Nullen
5 Alternierende Bits
6 Alternierende Bitpaare
7 Zufallsmuster

Spezielle Datenmuster: s. Speicherplatz 7032.
Das Datenmuster wird so spezifiziert, daß das 1. Byte die Spurnummer und die letzten 2 Bytes die Prüfsumme enthalten.

05
04 0000Y0 Testfolge
03

Mit Y=0 siehe Test 7
1 Nur schreiben
2 Schreiben/Lesen
3 Schreiben/Lesen/Vergleichen
4 Lesen/Vergleichen
5 Nur Lesen (CRC-Prüfung)
6 Schreiben/Lesen/Vergleichen, wobei beide Laufwerke abwechselnd angesprochen werden
7 Schreiben/Lesen/Vergleichen, nochmals Lesen/Vergleichen und Wiederholung desselben im Modus "Gelöschte Daten".

02
01 00000Z Spurfolge
00

Mit Z=0 Zufallsfolge
1 Erhöhen (Kopfbewegung von außen und innen)
2 Erniedrigen
3 Erhöhen/Erniedrigen
4 Springe zwischen innen und außen
5 Springe zwischen abnehmenden inneren und zunehmenden äußeren Spuren
6 Springe zwischen äußerster und abnehmender innerer Spur
7 Zufallsfolge

7020 Schalterregister (100310)

BIT	Oktal	Bedeutung bei BITXX=1
15	100000	Nur Anzeige der harten Fehler
14	040000	Nur Puffer leeren (Nur bei Rout. 01)
13	020000	Nur Puffer füllen (Nur bei Rout. 01)
12	010000	Kopf anheben bei jedem Test
09	001000	keine Wiederholung nach Fehlern
07	000200	keine Anzeige von Datenvergleichs-

		fehlern	
	06 000100	kein Kopfschütteln	
	03 000010	kein Initialisieren nach Fehlern	
	02 000004	Minidiskette 80 Spuren	
	00 000001	Unterdrückung des Fehlerprotokolls und erläuternden Texten	
7022	kleinste Spur (außen)		(000000)
7024	Größte Spur (innen)		(000114)
7026	Erster Sektor		(000001)
7030	Letzter Sektor		(000032)
7032	Spezielles Datenmuster Falls dieses Muster ungleich 0, wird dieser Wert anstelle der Standard-Generierung verwendet		(000000)
7034	Offset - Nur wirksam bei Rout. 03 (Formatierung). Der Offset gibt an, in welcher Reihenfolge (mit welchem Abstand) die Sektoren auf der Diskette nach dem Formatieren aufeinanderfolgen.		(000000)
7036	Steckeinheiten-Kennung. Stand.Wert 1. AFS Wert für 2. AFS: 173572		(173574)
7040	Adreßkennung. Standardwert 1. AFS: Wert für 2. AFS: 177202		(177172)
7042	Interruptvektor-Kennung Wert für 2. AFS: 070270		(070264)
7044	BDE-Kommando/Statusregisteradresse		(177502)

Die Generierparameter für die AFS des Konsolrechners sind im Abschnitt 2.14.3. enthalten.

2.13.10. Fehlerinterpretation

Bei auftretenden Fehlern erfolgt auf dem Bildschirm der Bedieneinheit eine Ausschrift, die den Fehler benennt und gleichzeitig angibt, bei welcher Spur- und Sektornummer, bei welchem Laufwerk und bei welchem Kommando der Fehler aufgetreten ist. Dies erfolgt durch eine Textzeile, z. B.

CRC = FEHLER SP = 35 SE = 14 LW = 1 KD = LS

Hierbei werden Spur- und Sektornummer dezimal angegeben. Als Kommando KD gelten:

KD=FP	Fülle Puffer
LP	Leere Puffer
SS	Schreibe Sektor
LS	Lese Sektor
SG	Schreibe gelöschte Daten
IN	Initialisieren

Nach dieser Textzeile erscheinen auf dem Bildschirm detaillierte Ausschriften in verschiedener Form (diese richtet sich nach der Fehlerart):

1. Form: (Fehler steht im Fehlerkoderegister)

```
SF GFLOPM  RTXX  AD  SW  C1  C2  C3  C4
                 C5  C6  C7  C8  PC  IV
   RT01    AFS-Puffer-Test
   RT02    Laufwerk-Test
   RT03    Formatieren
   AD      Kommando/Statusregisteradresse der AFS
   SW      Funktionscode
   C1      KSR-Ist-Inhalt
   C2      FSR-Ist-Inhalt
   C3      Beim Datenvergleich Zahl der fehlerhaften Bytes
   C4      Ohne Bedeutung

           Höherwertiger Teil     Niederwertiger Teil
   C5      Wortzähler             Fehlerkode
   C6      Akt. Spur LW1          Akt. Spur LW0
   C7      Zielsektor             Zielspur
   C8      Fehlerhafte Spur       Laufwerksstatus

   Der Laufwerksstatus beinhaltet:
   BIT
   07      LW-Anwahl (0=LW0;1=LW1)
   06      Schreibdichte LW1 (0=128byte/Sektor;1=256byte/Sektor)
   05      Zustand des Kopfes (0=Kopf abgehoben;1=Kopf angedrückt)
   04      Schreibdichte LW0 (0=128byte/Sektor;1=256byte/Sektor)
   03      Format von LW1 (0=FM;1=MFM)
   02      Format von LW0 (0=FM;1=MFM)
   01      Typ der angeschlossenen Folienspeichereinheit
   00      ermittelte Dichte (nur gültig nach Initialisieren,
           Statuslesen und Dichtesetzen)

2. Form: (Fehler steht im Fehlerstatusregister)

SF  GFLOPM  RTXX  AD  IS  SL  PS  PC  IV

   AD      Datenpufferregister-Adresse der AFS
   TS      FSR-Istinhalt
   SL      FSR-Sollinhalt

3. Form: (Fehler steht im Fehlerstatusregister)

YY  GFLOPM  RTXX   R0  R1  R2  R3  R4  R5
                       SP  PS  PC  IV
Für YY kann stehen:
   SF      Wortzählerüberlauf
   PA      Speicherparitätsfehler
   ZF      BUS-TIME-OUT-Fehler

   R0      Kommando/Statusregister-Adresse der AFS
   R1      Funktionskode
   R2      KSR-Ist-Inhalt
   R3      FSR-Ist-Inhalt
   R4      Busadreßzähler
   R5      Anfangsadresse des zu transportierenden Datenfeldes

4. Form: (Nach der Textzeile "Fehler beim Datenvergleich")

DF  GFLOPM  RTXX  AD  IS  SL  PS  PC  IV
```

RTXX	Siehe oben
AD	Byte-Nr.
IS	Istwert des Datenbytes
SL	Sollwert des Datenbytes

5. Form: (Nach der Textzeile "Daten-, aber kein CRC-Fehler")

DF GFLOPM RTXX 177777 IS SL PS PC IV

RTXX	Siehe oben
IS	Niederw. Byte: Spur-Nr. Höherwert. Byte: Sektor-Nr.
SL	Vergleichsfehlergesamtzahl

Welche der 5 Anzeigeformen verwendet wird, richtet sich nach der Fehlerart: Fehler, die im Fehlerkoderegister der AFS spezifiziert werden (z.B. Markensuchfehler, CRC-Fehler u.a.), führen zu Bildschirmausschriften der Form 1. Fehler, die im Fehlerstatusregister spezifiziert werden (z.B. Gelöschte-Daten-Fehler, BUS-TIME-OUT-Fehler), führen zu Bildschirmausschriften der Formen 2 und 3. Datenvergleichsfehler führen zu Bildschirmausschriften der Formen 4 und 5.

2.13.11. Funktionsbeschreibung

Das GFLOPM besteht aus sieben Routinen

RT01	AFS-Puffer-Test
RT02	Laufwerk-Test
RT03	Formatieren
RT04	Setze einf. Dichte
RT05	Setze dopp. Dichte
RT06	Doppeln vom LW0 zum LW1
RT07	Vergleichen der Disketten beider Laufwerke

RT01: Im AFS-Puffer-Test wird zuerst durch das Kommando "Fülle Puffer" der AFS-Puffer mit den Daten gefüllt, die in der Generierung als Muster angegeben sind. Die Daten werden so modifiziert, daß das 1. Byte die Spurnummer, das 2. Byte die Sektoradresse und die letzten 2 Byte die Prüfsumme enthalten. Danach wird durch das Kommando "Leere Puffer" der AFS-Puffer geleert, und die Daten werden verglichen. Bei generierter einfacher Aufzeichnungsdichte werden 128 Datenbytes in beiden Richtungen übertragen. Bei generierter doppelter Aufzeichnungsdichte werden 256 Datenbytes in beiden Richtungen übertragen. Vor der Übertragung der Daten zum AFS-Puffer wird die AFS initialisiert.

RT02: Nach der Initialisierung der AFS und Prüfung der Kennung der AFS werden durch das Programm je nach der Generierung verschiedene Teilroutinen zusammengefaßt zu Komplexen wie z. B.: "Schreiben", "Lesen", "Vergleichen". Diese Teilroutinen sind:

- Fülle Puffer
- Schreibe Sektor
- Lese Sektor
- Leere Puffer
- Datenvergleich
- Fehlerbehandlung
- Datenmusterberechnung
- Zufallszahlenberechnung
- Sektorberechnung

- Spurberechnung
- Laufwerkauswahl
- Laufwerkkommando-Aufbereitung
- Behandlung von gelöschten Daten
- Test auf Fertigbit
- Interruptbehandlung
- Textausgabe
- Ausgabe des Fehlerprotokolls

Die genannten Funktionen werden innerhalb generierbarer Spur- und Sektorgrenzen durchgeführt (s. Abschn. 2.13.9., Plätze 7022 ... 7030). Standardmäßig (d. h. wenn keine besonderen Generierungen angegeben werden, sondern nach Aufruf des Namens GFLOPM einfach gestartet wird) werden alle Spuren (0 ... 114 Oktal) und alle Sektoren (1 ... 32 Oktal) entsprechend FM, einfache Dichte, beim Testen erfaßt. Will man eine Diskette mit anderem Format testen, so ist die entsprechende Generierung (s. Abschn. 2.13.9.) zu beachten. Liegen die generierten Spur- bzw. Sektorgrenzen außerhalb der für das generierte Format gültigen Bereiche, so erfolgt nach der Textzeile "Spur/Sektorgrenzen unzulässig" der Programmabbruch. Standardmäßig arbeitet das Programm mit allen bereiten Laufwerken. Falls in der Testauswahl (s. Abschn. 2.13.9., Platz 7016) direkt angegeben, werden die Daten im Modus "Gelöschte Daten" behandelt, während standardmäßig kein Modus "Gelöschte Daten" angenommen wird. Ausnahme: Test 7.
Ohne spezielle Generierung wird das Datenmuster "Zufall" und die Testfolge 7 durchgeführt. Dies wird innerhalb eines Routinedurchlaufes im Modus "Gelöschte Daten" wiederholt. Die Routine gibt, falls erlaubt (Bit00=1 im Schalterregister, siehe Abschnitt 2.13.9., Platz 7020), nach einer Überschrift die bereiten Laufwerke sowie folgende Information aus:

- Gewählte Testfolge (TF=Y), Datenmuster (DM=X), Spurfolge (SF=Z)
  (s. Abschnitt 2.13.9., Platz 7016)
- Gewählte Routinezyklen (Zyklus - Zahl = ...) dezimal
- Gewählte Spur- und Sektorgrenzen dezimal
  Außerdem wird angegeben, welches Format und welche Aufzeichnungsdichte, sowie welcher Laufwerktyp (8" oder 5 1/4") generiert worden ist.
  Danach beginnt der eigentliche Testlauf.

Bei Standardgenerierung wird mit Laufwerk 0 begonnen, wenn nichts hinsichtlich der LW-Anwahl generiert ist. Entsprechend der gewünschten Spurfolge wird auf die erste Spur zugegriffen. Hier werden alle Sektoren innerhalb ihrer Grenzen bedient (z.B. beschrieben). An einem Schreibvorgang schließt sich ein "Lesen mit Datenvergleich" an (z.B. bei Testfolge 3), ohne daß die Spur zwischendurch gewechselt wird.
Erst wenn auf allen gewählten Sektoren einer Spur die Operation abgeschlossen ist, wird zur anderen Spur (bei Testfolge 6 zum anderen Laufwerk) übergegangen. Nachdem auf alle Spuren innerhalb der gewählten Grenzen zugegriffen wurde, wird zu anderen Laufwerk (falls bereit) übergegangen und dort wieder von vorn begonnen. Ein Durchlauf gilt als abgeschlossen, nachdem alle bereiten Laufwerke in den Test einbezogen waren.
Sollte die Routine mehrmals durchlaufen werden, so werden beim "Zufallsmuster" (und nur dort) die Daten neu berechnet. Alle Operationen sind mittels entsprechender Testfolge einzeln ausführbar (s. Abschn. 2.13.9., Platz 7016).
Tritt ein Fehler auf, so wird dieser für das Fehlerprotokoll registriert. Die sich anschließende Fehlerbehandlung kann durch das Schalterregister (s. Abschn. 2.13.9., Platz 7016) differenziert werden. Nach einem Fehler wird der Kopf auf eine andere Spur (+/-4) bewegt und "Lese Sektor" durchgeführt. Sollte auch dieses "Kopfschütteln" nicht fehlerfrei verlaufen, wird erst nach einem "Rücksetzen der AFS" der Kopf wieder zur Sollspur positioniert und die entsprechende Operation wiederholt, bevor auf den nächsten Sektor zugegriffen wird. Verläuft diese

Operation trotz 10-maliger Wiederholung in obiger Form nicht fehlerfrei, wird bei Generierung "Nur harte Fehler ausgeben" der Fehler registriert. Bei Generierung "Jeden Fehler ausgeben" erfolgt die Textausgabe nach jedem weichen Fehler.
Beim "Rücksetzen der AFS" (Initialisieren), was auch am Routinebeginn durchgeführt wird, werden Laufwerke und AFS in den Grundzustand gebracht. Sollte das trotz 10-maliger Aufforderung nicht möglich sein, bricht das Programm nach der Ausschrift "REKALIBRIERUNG UNMOEGLICH" seine Arbeit ab.
Wird während des Durchlaufes festgestellt, daß das aktuelle Laufwerk nicht mehr bereit ist ("Laufwerk 0/1 herausgenommen"), wird der Durchlauf mit dem anderen neu begonnen. Sind keine Laufwerke mehr bereit, wird nach der Textausschrift "ABBRUCH" die Steuerung an das Leitprogramm übergeben.
Nach jedem durch das Error-Bit ausgewiesenen Fehler wird über "Lese erweiterten Status" der Fehlerkode registriert.Die dabei gewonnene Information wird als Text über den Bildschirm ausgegeben.
Nachdem die Routine entsprechend Generierung x-mal durchlaufen wurde, gibt das Programm "Fehlerprotokoll" aus. Dieses ist unterdrückbar. Durch das Protokoll werden alle während des (der) Routinedurchlaufes (-Läufe) gesammelten statistischen Angaben über die Fehlerarten und die Gesamtzahl der Fehler (dezimal) ausgegeben. Danach folgt eine Zugriffstabelle folgender Form:

1. Spalte	2. Spalte	3. Spalte	4. Spalte	5. Spalte
SP-Nr., auf die	X-mal zugegriffen wurde wobei der Kopf	X-mal von einer anderen Sp. zu dieser hinbewegt werden mußte	Gesamtzahl der Fehler je Spur LW 0	Gesamtzahl der Fehler je Spur LW 1

Die beschriebenen Abläufe sind standardmäßig für FM-Format, einfache Dichte (8"-Disketten) enthalten. Diesselben Abläufe sind auch für DEC SPEZ und 8"-MFM sowie für Minidisketten (40- und 80-spurig) generierbar. Die Generierparameter für den Konsolrechner sind im Abschnitt 2.14.3. dargestellt.
Beim Format DEC SPEZ liefert das Programm GFLOPM die doppelte Wortzahl (128 Worte) für die Operationen Fülle Puffer/Leere Puffer und das entsprechende Kommando für die Laufwerkauswahl. Bei 8"-MFM unterscheidet GFLOPM zwischen Operationen auf Spur 0 und anderen Spuren. Spur 0 wird in einfacher Dichte, alle anderen Spuren werden in doppelter Dichte aufgezeichnet.
Bei Minidisketten werden alle Spuren einheitlich im MFM-Verfahren beschrieben.
Im Programm GFLOPM wird überprüft, ob die Spur/Sektorzahl im zulässigen Bereich (bei 8"-Laufwerken Spur 0 ... 76, bei Minidisketten Spur 0 ... 39 bzw. 0 ... 79) liegt. Als Standardgenerierung ist vorgesehen, daß zuerst RT01 und danach RT02 abläuft.

RT03: Nach der Initialisierung wird von GFLOPM das Schreibe-Sektor-Kommando mit einer je nach gewählter Dichte festen Pseudosektornummer ausgegeben. Die Spurfolge ist absteigend eingestellt. Die Sektorfolge wird an Hand des generierten Offset berechnet, alle Sektorennummern werden in der berechneten Reihenfolge zur AFS gesendet. Nachdem die AFS den letzten Sektor empfangen hat, beginnt das Mikroprogramm den Formatiervorgang.
Bei 8"-MFM wird in das Kommando das 8"-MFM-Bit eingefügt, Spur 0 wird mit einfacher Dichte, alle anderen Spuren werden mit doppelter Dichte formatiert.

RT04: Nach der Initialisierung wird von GFLOPM das Kommando "Setze Dichte" mit der Schlüsselnummer 111 ausgegeben. Das Dichtebit im Kommando (Bit 08) ist nicht gesetzt. Die AFS beginnt, in allen Sektoren und in allen Spuren Datenmarken für einfache Aufzeichnungsdichte zu schreiben.

RT05: Entspricht RT04; das Dichtebit im Kommando (Bit 08) ist gesetzt. Die AFS schreibt in allen Sektoren und in allen Spuren Datenmarken für doppelte Aufzeichnungsdichte.

Die Routinen RT04 und RT05 sind nur bei FM-Aufzeichnung anwendbar.

RT06: Diese Routine überträgt den Inhalt der generierten Spuren und Sektoren der Diskette des Laufwerkes 0 auf die entsprechenden der Diskette im Laufwerk 1. Dazu müssen beide Disketten das gleiche Format haben. Fehler beim Lesen der Diskette im Laufwerk 0 bzw. beim Beschreiben der Diskette im Laufwerk 1 werden angezeigt.

RT07: Voraussetzung ist, daß die Disketten in den beiden Laufwerken gleich formatiert sind. Die Routine vergleicht den Inhalt der generierten Spuren und Sektoren der Diskette im Laufwerk 0 mit jenem der Diskette des Laufwerkes 1, wobei der Disketteninhalt des LW0 als Sollwert und jener des LW1 als Istwert behandelt und bei Unstimmigkeit protokolliert werden. Fehler beim Lesen der Disketten werden ebenfalls angezeigt.

ACHTUNG! Vor Start des Prüfprogramms GFLOPM ist die PSU- Diskette gegen entsprechende Arbeitsdisketten zu tauschen.

2.14. Konsolprüfprogrammdiskette DS0

Die Prüfprogrammdiskette DS0 hat die Dokumentationsnummer 1.57.700100.2

2.14.1. Leitprogramm LPROEM

Das Leitprogramm LPROEM für die Prüfprogramme des Konsolrechners liegt auf der Diskette DS0 als sich selbstladend und organisierend vor. Durch den Anfangslader des CIB wird über Konsolrechner-RESET bzw. im Bedienzustand des Konsolrechners mit 173000G der Absolutlader (Spur 1, Sektor 1) der PSU-Diskette DS0 eingelesen, der wiederum das Leitprogramm LPROEM lädt und startet. Nach Meldung von LPROEM mit

```
LPROEM KSS VERS. n - 87
BEDIENUNG KMBE [KS] - FSE [FS]
ZE  K1620
OPERATIVSPEICHER = 000000  137776
PSU-SPEICHERBEREICH  065400  137776;
```

quittiert man mit <CR>. Auf die Geräteanforderung

GERAETEKONFIGURATION:

muß "FS" und bezüglich der Laufwerksspezifikation

DISKETTE LAUFWERK (0\1);

ist mit der entsprechenden Laufwerksnummer (Standard: DS0 im LW0 —> 0) zu quittieren. Dadurch wird die Folienspeichereinheit initialisiert, die Laufwerksbereitschaft geprüft, das Inhaltsverzeichnis und die Verbindungszellen für die einzelnen auf der Diskette DS0 vorhandenen Prüfprogramme geladen. Das Leitprogramm und die Konfiguration sind nach Rückmeldung mit ":" arbeitsfähig.
Durch <CR> wird die Bedienungsanleitung angeboten. Nach Anzeige bzw. Ablehnung

der Bedienungsanleitung meldet sich LPROEM mit dem Prompt " > ".
Bei Neustart des im Konsolrechner befindlichen LPROEM über 20000G aus dem Bedienzustand wird die LPROEM-Meldung analog durchlaufen, jedoch ist die Bedienungsanleitung durch vormalige Arbeit im LPROEM überschrieben. Nach erfolgreicher Initialisierung der Folienspeichereinheit erscheint nach <CR> das Promptzeichen " > ".
Bei Wiederstart über 20026G aus dem Bedienzustand des Konsolrechners wird die LPROEM-Meldung nicht durchlaufen, da die eingestellte Konfiguration und die eventuell geladene Generierung erhalten bleiben. Die Bedienungsanleitung ist nicht verfügbar. Es wird lediglich das Laufwerk initialisiert und LPROEM meldet sich mit " > ".

2.14.2. Prüfprogramme für den Konsolrechner

Entsprechend der Gerätekonfiguration des Konsolrechners sind zu seiner Testung die in Tabelle 2-1 (siehe 2.1.1.) aufgeführten Prüfprogramme notwendig und auf DS0 hintereinander vorhanden. Mit LTKONF werden sie angezeigt:

```
>LTKONF
 ZE  K1620
OPERATIVSPEICHER = 000000 137776
(hier wird die aktuelle PSU-Speicherbereichsbelegung angezeigt)

KONFIGURATION DISKETTE LAUFWERK (0\1) 0
ZVZW20 AUSG.00 002252 ZARI20 AUSG.00 001720 ZSHI20 AUSG.00 001562
ZBBI20 AUSG.00 004170 ZBEF30 AUSG.01 005426 ZADRAR AUSG.00 001116
ZSPE20 AUSG.00 002442 ZTRAPI AUSG.00 007726 ZBUSCI AUSG.02 004302
AKSAIS AUSG.04 014260 GFLOPM AUSG.01 251166
DISKETTE BELEGT BIS (SPUR; SEKTOR)=000iii; 0000jj
```

Die Prüfprogramme können im Einzel- oder Dauertest abgearbeitet werden (notwendige Prüfhilfsmittel müssen dazu angebracht sein).

Einzeltest:

- Aufrufen des Prüfprogramms mit Hilfe seines Namens
- Generierung der vom Standard abweichenden Parameter
  . bei Arbeit mit dem Leitprogramm über PS-, RS-, PZ-, RZ- und GN-Eingaben entsprechend der Parameterreihenfolge
  . bei Arbeit ohne Leitprogramm nach Programmstart "G" auf Ladeadresse +14 und automatischem Stopp im Generierhalt (Ladeadresse +22) durch Beschreiben der entsprechenden Speicherzellen im Bedien-Halt-Zustand des Konsolrechners
- Abarbeiten des Prüfprogramms
  . bei Arbeit mit dem Leitprogramm durch die Kommandos "QT SH" bzw. "QT ST"
  . bei Arbeit ohne Leitprogramm durch das Kommando "P"

ACHTUNG! Vor Start des GFLOPM die PSU-Diskette herausnehmen und Testdisketten bzw Arbeitsdisketten einlegen.

Dauertest:

- Einstellen einer Dauertestgenerierung über "DAUGEN" (siehe Pkt. 2.2.) bzw. Aufruf einer vorhandenen Dauertestgenerierung über "DGNEIN NR = n", n = 1, ..., '20;
- Start der Dauertestgenerierung mittels "DAUTST QT ST" bzw. "DAUTST QT SH"

ACHTUNG! Bei einer Dauertestung mittels des Prüfprogramms GFLOPM ist dieses in den Operativspeicher des Konsolrechners zu laden, bevor die Dauertestgenerierung eingelesen wird. Vor dem Start der Dauertestgenerierung ist die PSU-Diskette durch eine Testdiskette zu ersetzen.

2.14.3. Wichtige Generierungen für den Konsolrechner

Die nachfolgend aufgeführten Generierungen für die einzelnen Prüfprogramme und Baugruppen ermöglichen eine sinnvolle Testung des Konsolrechners. Die Dauertestgenerierungen sind auf der Diskette DS0 unter der angegebenen Generiernummer vorhanden.

2.14.3.1. Dauertestgenerierung NR.= 1

Die auf der Diskette DS0 vorhandene Dauertestgenerierung NR.=1 hat die Dokumentationsnummer 1.57.700250.1 .
Diese Generierung umfaßt die Prüfprogramme der ZVE-Testung sowie die Programme der Unterbrechungs- und Bussteuerungstestung. Diese Programme sind mit ihren Standardparametern in die Dauertestgenerierung aufgenommen, lediglich bei ZBUSCI sind nur die Routinen 01 und 02 generiert (Routine 03 CIB-ROM-Prüfung ist gesondert generiert):

```
>DAUGEN
>>ZVZW20 ( )
>>ZARI20 ( )
>>ZSHI20 ( )
>>ZBBI20 ( )
>>ZBEF30 ( )
>>ZADRAR ( )
>>ZTRAPI ( )
>>ZBUSCI (,3 <CR>

>DGNAUS NR. = 1
```

Die mit DGNEIN NR. = 1 gerufene Generierung kann mit DGNANZ angezeigt werden:

```
>DGNEIN NR. = 1

>DGNANZ
:AN
NR=000001  ZVZW20
000000
NR=000002  ZARI20
000000
NR=000003  ZSHI20
000000
NR=000004  ZBBI20
000000
NR=000005  ZBEF30
000000
NR=000006  ZADRAR
000000
NR=000007  ZTRAPI
000000
NR=000010  ZBUSCI
000000 000003
```

Diese mit DAUTST QT SH bzw. ST gestartete Dauertestung benötigt pro Durchlauf ca. 8 min, und ist in ihrem Endloslauf mit Unterbrechungstaste "BREAK" zu stoppen. Das Leitprogramm ist über 20000G oder 20026G erneut startbar.
Im ZTRAPI sollte man die Unterbrechung vermeiden, um sich die RESTART-Möglichkeit zu erhalten und nicht an den verstellten PSW- und SP-Werten zu scheitern.

2.14.3.2. Dauertestgenerierung NR.= 2

Die auf der Diskette DS0 vorhandene Dauertestgenerierung NR.=2 hat die Dokumentationsnummer 1.57.700251.8
Diese Generierung prüft mit dem Prüfprogramm ZSPE20 den freien Lese/Schreibspeicher. Damit möglichst ein großer Bereich direkt geprüft wird, ist folgender Ablauf einzuhalten:

- Neustart, d.h. DS0 einlesen
- DGNEIN NR.=2 rufen
- DAUTST QT SH

Durch diesen Ablauf wird ZSPE20 gleich in Anschluß an das Leitprogramm auf die 65400 geladen. Da ZSPE20 den Speicher bis zur 70042 belegt, kann ab 70044 bis zum CIB-ROM (137776) der Speicherbereich getestet werden. Für den unteren Bereich gilt das unter 2.1. gesagte. Die Dauertestgenerierung ist dann:

```
>DAUGEN
>>ZSPE20 (,3,,,70044,137776,0,0,0 <CR>

>DGNAUS NR. = 2
```

Die gerufene Dauertestgenerierung wird dann wie folgt angezeigt:

```
>DGNEIN NR. = 2

>DGNANZ
:AN
NR = 000001  ZSPE20
000000 000003 000000 000000 070044 137776 000000 000000
000000
```

D.h. standardmäßig gebliebene Parameter innerhalb der Reihenfolge werden mit 000000 angezeigt. Nach dem letzten geänderten Parameter wird die Generieranzeige abgebrochen, siehe dazu auch die anderen Generieranzeigen.

2.14.3.3. Speichereinzeltestgenerierung

Der in Pkt. 2.1.2.2. und Pkt. 2.9.11. angesprochene Einzeltest ohne Leitprogramm mit ZSPE20 muß wie folgt generiert werden:

```
7004/100002
7006/000003
7010/000000
7016/007432
7020/65376
7022/70044
7024/137776
7026/000001
```

Dabei wird ZSPE20 als auf 65400 geladen vorausgesetzt.

2.14.3.4. Dauertestgenerierung NR.= 3

Die auf der Diskette DS0 vorhandene Dauertestgenerierung NR.=3 hat die Dokumentationsnummer 1.57.700252.6 .
Diese Generierung prüft mittels des Prüfprogramms AKSAIS die Anschlußsteuerung des seriellen Interfaces. Es ist generiert:

```
>DAUGEN
>>AKSAIS(,,,,60001,177570,177500,,,,,,,,,100 <CR>

>DGNAUS NR. = 3
```

Die Anwendung der Generierung setzt voraus, daß die Kurzschlußeinrichtungen an den griffseitigen Steckverbindern (STE 051-2000 am freien IFSS-Kanal und 051-2290 am S2-Kanal) der AIS-STE 051-1760 vorhanden sind. Die eingelesene und angezeigte Dauertestgenerierung ist:

```
>DGNEIN NR. = 3

>DGNANZ
:AN
NR = 000001  AKSAIS
000000 000000 000000 000000 060001 177570 177500 000000
000000 000000 000000 000000 000000 000000 000000 000100
```

2.14.3.5. Dauertestgenerierung NR.= 4

Die auf der Diskette DS0 vorhandene Dauertestgenerierung NR.=4 hat die Dokumentationsnummer 1.57.700253.4 .
Diese Generierung ermöglicht mit Hilfe des Prüfprogramms GFLOPM die Testung der Anschlußsteuerung des Foliensperichers und

a) unter Voraussetzung einer physisch einwandfreien, entsprechend formatierten Diskette die Testung des Laufwerkes und seiner Zuverlässigkeit bzw.
b) unter Vorliegen eines geprüften Laufwerkes die Testung von Disketten und deren Datenerhalt.

Die Generierung entsprechend a) für beide Laufwerke über je alle Spuren und einen Sektor lautet:

```
>DAUGEN
>>GFLOPM (,3,,,164531,4,0,117,17,17 <CR>

>DGNAUS NR. = 4
```

Die Anzeige der eingelesenen Generierung lautet:

```
>DGNEIN NR. = 4

>DGNANZ
:AN
NR = 000001  GFLOPM
000000 000003 000000 000000 164531 000004 000000 000117
000017 000017
```

Die Generierung für b) ist analog a) jedoch über alle Sektoren , d.h. statt 17,

17 muß 1,32 gegeben werden.

Um sich die Diskette DS0 nicht zu zerstören, muß unbedingt folgender Ablauf eingehalten werden:

- Aufruf von GFLOPM, d. h. Laden des Programms in den Operativspeicher des Konsolrechners
- DGNEIN NR. = 4 rufen
- DS0 aus Laufwerk nehmen und Testdisketten einlegen
- DAUTST QT SH bzw. ST

2.14.3.6. Formatiergenerierungen

Mit dem Prüfprogramm GFLOPM ist auch das Formatieren von Disketten möglich; der Ablauf dazu ist:

- GFLOPM rufen
  >GFLOPM
- nach Quittung mit Doppelpunkt Diskette DS0 aus Laufwerk 0 nehmen und zu formatierende Diskette einlegen.
- Formatiergenerierung eingeben

  . 5 1/4"-MFM-26 Sektoren, LW0:

  >GFLOPM: GN=64000,4,0,117,1,32,,*E: RS=4E

  . 5 1/4"-MFM-26 Sektoren, LW1:

  >GFLOPM: GN=124000,4,0,117,1,32,,*E: RS=4E

  Bei * kann der Offset als Oktalzahl eingegeben werden. Standard ist 0, d.h. es werden die Sektoren in aufsteigender Nummerierung hintereinander angelegt. Bei einem anderen Offset werden die Sektoren mit Nummerierungssprüngen aufgebaut, z. B.
  Offset 15 : 1,17,2,20,3,21 usw.
  Die Disketten für DS0 sollten mit Offset 10, die Disketten für DSi, i > 0, mit Offset 15 formatiert werden; damit wird ein zeitoptimaler Zugriff ermöglicht.

- Start des Programms durch die Fortsetzung der Kommandos mit QT und SH bzw. ST
- Der Formatiervorgang wird auf dem Bildschirm protokolliert.

2.14.3.7. Dauertestgenerierung NR.= 5

Die auf Diskette DS0 vorhandene Dauertestgenerierung NR.=5 hat die Dokumentationsnummer 1.57.700254.2 .
Diese Generierung prüft mit dem Prüfprogramm ZBUSCI, Routine 03, die Prüfsummen der CIB-ROMs. Zu diesem Zweck ist generiert:

```
>DGNEIN NR. = 5

>DGNANZ
:AN
NR = 000001  ZBUSCI
000000 000004
```

Bei auftretendem Fehler ist über die im Abschnitt 2.11. beschriebene Zuordnung der fehlerhafte ROM lokalisierbar.
Bemerkung: Diese Generierung erfordert ON-LINE-Betrieb des Konsolrechners (Kopplung zwischen K 1840 und Konsolrechners)

2.14.3.8. Dauertestgenerierung NR.= 6

Diese auf der Diskette DS0 vorhandene Dauertestgenerierung NR.=6 hat die Dokumentationsnummer 1.57.700255.0 .
Während die anderen Generierungen zur Prüfung einzelner Baugruppen gedacht sind, kann mit dieser Generierung ein Dauertest des Konsolrechners durchgeführt werden.
Es ist generiert:

```
>DGNEIN NR. = 6

>DGNANZ
:AN
NR = 000001  ZVZW20
000000
NR = 000002  ZARI20
000000
NR = 000003  ZSHI20
000000
NR = 000004  ZBBI20
000000
NR = 000005  ZBEF30
000000
NR = 000006  ZADRAR
000000
NR = 000007  ZTRAPI
000000
NR = 000010  ZBUSCI
000000
NR = 000011  AKSAIS
000000 000000 000000 000000 060001 177570 177500 000000
000000 000000 000000 000000 000000 000000 000000 000100
```

ZBUSCI erfordert für Routine 03 ON-LINE-Betrieb des Konsolrechners. AKSAIS setzt die Kurzschlußeinrichtungen (lt. 2.1.2.3.) voraus.

2.14.3.9. Generierung für das Doppeln von Disketten

Das Doppeln von Disketten ist mit dem Prüfprogramm GFLOPM möglich. Voraussetzung ist, daß die Zieldiskette auf das gleiche Format wie die Qelldiskette formatiert ist. Der Ablauf ist:

- GFLOPM von Diskette DS0 rufen (>GFLOPM)
- nach Quittung mit Doppelpunkt in das Laufwerk 0 die Quelldiskette und in das Laufwerk 1 die Zieldiskette einlegen
- Generierung eingeben
  GN=164151,4,0,117,1,32E:
- Routinesteuerwort eingeben
  RS=40E:
- Starten des Doppelns mit QT: ST

Die Übertragung nur bestimmter, generierter Spuren und Sektoren ist möglich.

2.14.3.10. Generierung für das Vergleichen von Disketten

Das Vergleichen von Disketten gleichen Formats ist mit dem Prüfprogramm GFLOPM möglich, der Ablauf dazu ist:

- GFLOPM in den Speicher laden (>GFLOPM:)
- die zuvergleichenden Disketten in die Laufwerke geben
- Generierung eingeben
  GN=164141,4,0,117,1,32E:
- Routinesteuerwort eingeben
  RS=100E:
- Starten des Vergleichens mit QT: ST

Das Vergleichen nur bestimmter, generierter Spuren und Sektoren ist möglich.

3. Konsolsteuerung und Bootstrapping

3.1. Zusammenfassung

In diesem Kapitel sollen der Aufbau des Konsolprogramms (Dokumentationsnr. 1.57.700300.4), seine wesentlichsten Aufgaben beschrieben und Erläuterungen zu allen wichtigen Konsolkommandos gegeben werden, die u.a. die Systemdiagnose auf verschiedenen Ebenen unterstützen und den Betriebssystemstart veranlassen.

3.2. Voraussetzungen

Als minimale Hardware-Voraussetzungen sind zunächst

- das Konsolsubsystem (KSS) mit

  . ZVE K 1620
  . Hauptspeicher K 3564
  . AFS K 5161
  . AIS K 8060.02
  . 5 1/4"-Floppy-Disk-Laufwerk
  . BDE K 8911.80

  sowie

- das Konsolinterface CIB

  erforderlich.

Darüberhinaus sind in Abhängigkeit der durchzuführenden Maßnahmen natürlich weitere Hardware-Bestandteile des Rechnersystems K 1840 notwendig.
Neben den hardwareseitigen Bedingungen müssen die entsprechenden 5 1/4"-Disketten mit einem ladbaren System (Standard-Konsoldiskette DS1, Dokumentationsnr. 1.57.700101.0) sowie gegebenenfalls weitere Testdisketten vorhanden sein.

3.3. Bedienungshinweise

Das Laden des Operationssystems des Konsolrechners K 1620 (Konsolprogramm CONSOL.SYS) von der Standardkonsoldiskette DS1 erfolgt

- automatisch
  . beim Netzeinschalten
  . beim kurzfristigen Netzausfall.

- aus dem Bedienzustand des K 1620 mit
  . D

  . 173000G

  . 140000G

  . 140200G

  . R7/xxxxxx 141330
    P

  Die ersten vier Möglichkeiten entsprechen dabei dem Kaltstart, die letztere (141330G) dem Warmstart des Systems.

- mit dem Konsolkommando REBOOT.

Die Standardkonsoldiskette DS1 kann dabei in eines der beiden Laufwerke eingelegt werden, der Anfangslader versucht zunächst, vom Laufwerk 0 (DRIVE_0) zu laden und danach vom Laufwerk 1 (DRIVE_1). Beim Booten des Konsolprogramms wird die aktuelle Konfiguration (welche Laufwerke sind angeschlossen, wo liegt die Konsoldiskette) bestimmt und bis zu einem Neuladen abgespeichert.

Mittels des vorhandenen Boot-Blockes auf der Standardkonsoldiskette DS1 wird das Konsolprogramm in den Speicher des Konsolrechners geladen und gestartet. Die Meldung des Konsolprogramms erfolgt mit:

```
BOOTING FROM DRIVE_x

        CPU HALTED,SOMM CLEAR,STEP=NONE,CLOCK=NORM
        RAD=HEX,ADD=PHYS,DAT=LONG,FILL=00,REL=00000000
        (Zustand der Laufwerke 0 und 1)
        INIT SEQ DONE
        HALTED AT 00000000

        (RELOADING WCS)
        LOAD DONE, XX00 MICROWORDS LOADED
        VER: PCS=01 WCS=0E-10 FPLA=0E CON=V01-00-L

>>>
```

Folgende Fehlermeldungen können beim Laden des Systems auftreten:

- ?FLOPPY NOT READY
- ?FLOPPY NOT READY DRIVE x
- ?NO BOOT ON FLOPPY DRIVE_x
- ?FLOPPY ERROR ON BOOT
- ?B-I/O ERROR

Sie zeigen an, daß die Laufwerke nicht bereit bzw. nicht angeschlossen sind, daß sich kein Bootblock auf der/den Disketten befindet oder daß bei existierendem Bootblock Fehler bei der Abarbeitung bzw. beim Laden des Systems auftreten. Je nach Fehlerart ist entsprechend das Laufwerk anzuschließen oder zu schließen oder die Diskette zu wechseln.

Nachfolgend sollen vier typische Beispiele erläutert werden:

(1) Ausgangspunkt:
Im Laufwerk 0 liegt die gültige Konsoldiskette.

```
BOOTING FROM DRIVE_0
```

Nach Abschluß des Bootens kann mit einem Laufwerk gearbeitet werden. Es erfolgt keine spezielle Information über das Laufwerk 1; erst nach dem Kommando ENABLE DUAL erfolgt eventuell die Mitteilung NO DRIVE_1, falls nur ein Laufwerk vorhanden ist.

(2) Ausgangspunkt:
Laufwerk 0 physisch nicht vorhanden, offen oder ohne Diskette, Laufwerk 1 enthält die gültige Konsoldiskette.

```
?FLOPPY NOT READY DRIVE_0
BOOTING FROM DRIVE_1
```

Es wird darüber informiert, daß vom Laufwerk 0 nicht gelesen werden kann. Nach Abschluß des Bootens kann ohne Einschränkung so gearbeitet werden wie im Fall (1). Die Tatsache, daß DS1 jetzt auf dem Laufwerk 1 liegt, wird vom Konsolprogramm automatisch berücksichtigt. Das eventuell abgeklemmte Laufwerk 0 fehlt logisch so lange nicht, bis man versucht, mittels ENABLE DUAL auf dieses zuzugreifen.

(3) Ausgangspunkt:
Im Laufwerk 0 liegt eine Diskette ohne gültigen Bootblock, im Laufwerk 1 die gültige Konsoldiskette.

```
?NO BOOT ON FLOPPY DRIVE_0
BOOTING FROM DRIVE_1
```

Die Arbeitsweise ist analog (2).

(4) Ausgangspunkt:
Beide Laufwerke nicht angeschlossen, offen oder ohne Diskette.

```
?FLOPPY NOT READY DRIVE_0
?FLOPPY NOT READY DRIVE_1
```

Durch Eingabe eines beliebigen Zeichens über die Tastatur wird die Suche auf den Laufwerken beliebig oft wiederholt bis auf einem Laufwerk eine bootbare Diskette gefunden wird.

Um dem Bediener jederzeit den vollen Zustand des Konsolprogramms sowie der Floppylaufwerke anzeigen zu können, ist das SHOW-Kommando einzugeben. Zu jedem Laufwerk werden dann folgende Angaben gemacht:

NO DRIVE_x	Das Laufwerk x war zum Zeitpunkt des Bootens physisch abgeklemmt. Nach seiner Installation ist ein Neubooten erforderlich, um dessen Vorhandensein zu markieren.
DRIVE_x=DIS	Das Laufwerk x ist physisch vorhanden, es wurde aber logisch noch nicht zugewiesen und ist deshalb nicht ansprechbar.

DRIVE_x=DXy	Das Laufwerk x ist logisch unter dem Namen DXy: ansprechbar.
DRIVE_x=CONS	Auf dem Laufwerk x liegt die Konsoldiskette DS1. Diese kann uneingeschränkt so angesprochen werden, als wäre nur dieses Laufwerk vorhanden. (ohne Angabe von DX0: oder DX1:)
DRIVE_x=CSAx	Diese Angabe erfolgt nur, wenn das K 1840-Betriebssystem gebootet worden ist. Sie zeigt an, welches Laufwerk unter der Gerätebezeichnung CSA1: oder CSA0: angesprochen werden kann.

Nach dem erfolgreichen Laden des Konsolprogramms befindet sich das System im Kommandoeingabemodus und es können alle in 3.6. beschriebenen Kommandos vom Nutzer verwendet werden. Der Zugriff zum zweiten Floppy-Laufwerk ist vorerst blokkiert. Kurzbeschreibungen aller Kommandos, der möglichen dazugehörigen Abkürzungen sowie der Fehlermeldungen sind in entsprechenden Help-Files enthalten. Diese können mit

>>>HELP<CR>	für die Kommandos
>>>HE ABBREV<CR>	für die Abkürzungen
>>>HE ERROR<CR>	für die Fehlermeldungen
>>>HE BOOT<CR>	für Boot-Parameter

auf dem Konsolterminal ausgegeben werden.

3.4. Funktionsbeschreibung und Aufbau

Das Minimalbetriebssystem gliedert sich in zwei Teile: einen ROM-residenten Programmteil von ca. 6 kB Größe, der sich auf dem Konsolinterface (CIB) befindet (Adressbereich %O140000...%O157777) und einen nichtresidenten Teil, der sich als Programmfile CONSOL.SYS auf der Standard-Konsoldiskette DS1 befindet (Adressbereich %O0...%O43777).

Der ROM-residente Teil umfaßt u.a.:

- Testprogrammfolgen für grundlegende Befehlstests des Konsolrechners sowie einen Speichertest
- Serviceroutinen (Driver) für die Ansteuerung der Bedieneinheit (Initialisierung, Lesen, Schreiben, usw.)
- Serviceroutinen (Driver) für die Ansteuerung des 5 1/4"-Floppy-Laufwerkes einschließlich der dazugehörigen Fehlermeldungen (Fileeröffnung, Lesen, Schreiben, usw.)
- Sämtliche EMT-Behandlungsroutinen
- Serviceroutinen für Konvertierungen.

Das Konsolprogramm enthält u.a.

- Alle Routinen zur Kommandoeingabe, -auswertung und -ausführung einschließlich der dazu notwendigen Analysetabellen

- alle für die Kommunikation und Parameterübermittlung zu den weiteren Testmonitoren notwendigen globalen Speicherdefinitionen.

Nach dem Start auf der Adresse %0173000 wird ein Sprung auf die Adresse %0140000 im ROM-residenten Teil des Systems ausgeführt. Danach erfolgt die Initialisierung der Bedieneinheit BDE K 8911.80 . Im Anschluß daran wird ein Test einer Reihe von K 1600-Maschinenbefehlen bzw. von Modifikationen derselben vorgenommen. Damit soll die korrekte Arbeit des Konsolprogramms und der Monitore weitgehend gesichert werden. Diese Programmteile enden im Fehlerfall stets mit einem HALT-Befehl; an Hand des dabei angezeigten Programmzählerstandes kann eine Ortung des Fehlers vorgenommen werden.

Im Anschluß an das Initialisieren der AFS K 5161 erfolgt das Laden des Boot-Blockes von der Standard-Konsoldiskette, durch dessen Abarbeitung dann das gesamte Konsolprogramm in den Speicher des Konsolrechners geladen wird. Nach weiteren vorbereitenden Maßnahmen (u.a. Laden der Interruptvektoren, Initialisierung von Speicherbereichen) und der Meldung ">>>" ist die Bedieneinheit für die Kommunikation mit dem Bediener bereit.

3.5. Kommandobeschreibung

3.5.1. Allgemeines

Unter <adr> bzw. <dat> wird im folgenden eine Zahl <zahl> (oktal oder hexadezimal) bzw. eine symbolische Adressangabe (nur bei EXAMINE und DEPOSIT) verstanden:

R0, R1, ..., R11, R12, R13, R14, R15, AP, FP, SP, PC	CPU-Register
PSL	Prozessorstatuslangwort
*	Zuletzt benutzte Adresse
+	Nächste Adresse
-	Vorhergehende Adresse
@	Letzter Datenwert in EXAMINE oder DEPOSIT als Adresse
<zahl>	Zeichenfolge im augenblicklich eingestellten Zahlensystem oder durch vorangestelltes %O als Oktalzahl bzw. durch %X als Hexadezimalzahl
<filespec>=[<device>]<filename>	vollständige Filespezifikation
<device>	Geräteangabe; DX0: oder DX1:. Wird <device> nicht angegeben, erfolgt Zugriff zur Konsoldiskette.

<filename> Filename einschließlich Filetyp

Die in den nachfolgenden Überschriften enthaltenen Abkürzungen sind die minimal zur Erkennung der Kommandos notwendigen Zeichenfolgen, z.B. "E" für EXAMINE. Werden mehr als diese notwendigen Zeichen eingegeben - maximal bis zur vollen Bezeichnung -, müssen sie jedoch exakt mit der Kommandobezeichnung bis zur entsprechenden Position übereinstimmen. Alle weiteren Zeichen bis zum Erreichen eines Trennzeichens (Leerzeichen, Tabulator, usw.) werden übergangen. Das gleiche gilt für weitere Kommandoungergliederungen, Schalter, symbolische Adressangaben usw. Diese optionalen Teile werden jeweils in eckige Klammern eingeschlossen.
Als Trennzeichen zwischen den einzelnen Kommandobestandteilen muß mindestens ein Leerzeichen oder Horizontaltabulator stehen. Nicht notwendig sind diese Trennzeichen bei den Teilen eines Kommandos, die mit einem Sonderzeichen, z.B. "/" oder ":", beginnen.
Alle Kommandos werden mit <CR> abgeschlossen.

Im weiteren erfolgt eine Beschreibung aller Konsolkommandos in alphabetischer Reihenfolge.

3.5.2. BOOT (B)

Format: B [<device>][<bootdev>]

Funkt.: Das BOOT-Kommando veranlaßt die Abarbeitung einer Bootstrap-Folge. Dabei werden verschiedene Systemgeräte unterstützt.
<device> ist dabei der logische Name des Floppylaufwerkes, von dem der Indirektkommandofile gelesen werden soll. Wird <device> nicht angegeben erfolgt der Zugriff zur Konsoldiskette.
Der Gerätename <bootdev> hat das Format DDn, wobei DD eine aus zwei Buchstaben bestehende Gerätebezeichnung und n eine Gerätenummer (eine Ziffer) ist. Wird kein Gerät <bootdev> angegeben, so erfolgt das Booten vom Standardsystemgerät, wobei die Indirekt-Kommandodatei DEFBOO.CMD verarbeitet wird. Sie enthält die notwendigen Konsolkommandos, um das Booten vom Standardgerät zu veranlassen. Bei Angabe einer Gerätebezeichnung DDn wird durch das Konsolprogramm das Indirektkommandofile DDnBOO.CMD auf der Standardkonsoldiskette gesucht und abgearbeitet (identisch mit @DDnBOO.CMD), das die entsprechenden Kommandofolgen zum Booten eines Systems vom jeweiligen Gerät enthält. Einzelheiten über die Bedeutung der verschiedenen Parameter sind im Helpfile BOOT.HLP enthalten.

Bei Nichtangabe von n wird standardmäßig die Gerätenummer 0 angenommen.

Bei Eingabe von Sx (z.B. SR, SB) als Gerätebezeichnung wird das Laden des Diagnose-Supervisors veranlaßt.

Nach dem Laden der entsprechenden Programme (Supervisor bzw. Betriebssystem) und deren Start erfolgen die jeweiligen Meldungen auf dem Konsolterminal, das sich damit im Programm-E/A-Modus befindet.

Als Spezialfall existieren zu ausgewählten Geräten noch weitere Kommandofiles auf der Standard-Konsoldiskette:

DDxBOO.CMD, x=A,B:In diesem Fall ist vor dem Start der Boot-Datei mit

>>>DEPOSIT R3 x

in das CPU-Register R3 die Geräteeinheitennummer, von der ein System geladen werden soll, einzutragen. Das ist nur für solche Varianten von Bedeutung, in denen das File DDxBOO.CMD nicht explizit existiert.

DDnGEN: Im Falle des Aufrufs

>>>@DDnGEN

erfolgt ein Laden vom Gerät DDn mit der Möglichkeit, Systemparameter während des Ladens durch Anhalten im Programm SYSBOOT zu verändern (Steuerung durch Boot-Steuerflags).

Bsp.: >>>B DR1 Abarbeitung der Kommandodatei DR1BOO.CMD und Start des dazugehörigen Systems.

Bemerkung: Vor dem Aufruf eines Boot-Kommandofiles sind stets die Standardwerte HEX, LONG und PHYS einzustellen, anderenfalls erfolgen während des Ladens Fehlermeldungen infolge illegaler Zahlendarstellungen!

Während des Ladens des Betriebssytems oder des Diagnose-Supervisors von einem Plattengerät wird ein ROM-residentes Programm (ab Adresse %X20003000) abgearbeitet. Dieser ROM befindet sich auf der BLP 8811 (MSB). Mit diesem Programm wird ein kurzer Befehlstest durchgefuehrt. Danach werden die im System vorhandenen Speichersubsysteme auf korrekte Konfiguration getestet sowie ein Arbeitsspeicher von 64 kB Größe gesucht, der für das Laden des Programms VMB.EXE benötigt wird. In einem Fehlerfall wird eine Fehlerausschrift auf dem Konsolterminal ausgegeben. Das Programm und das Kommandofile sind anschließend mit ^P und ^C abzubrechen.

3.5.3. CLEAR (CL)

Format: CL SO[MM]

Funkt.: Die "STOP ON MICROBREAK MATCH" (SOMM)-Erlaubnis auf dem Konsolinterface (CIB) wird gelöscht.

Format: CL S[TEP]

Funkt.: Jeder eingestellte Taktschritt-Modus (Einzelbuszyklus, Einzelzeitzyklus, Einzelbefehlszyklus) wird gelöscht und die CPU arbeitet im normalen (free running) Modus.

3.5.4. CONTINUE (C)

Format: C

Funkt.: Start der Abarbeitung eines Programms im Hauptspeicher an der Adresse, die momentan im PC enthalten ist. Es wird keine Initialisierung der CPU vorgenommen. Das Konsolterminal geht in den Programm-E/A-Modus über.

3.5.5. DEPOSIT (D)

Format: D[/sw] <adr> <dat>

Funkt.: Beschreiben der Speicherzelle <adr> mit dem Wert <dat>. Durch Angabe zusätzlicher Schalter nach dem Kommando können spezielle Adressbereiche, Datendarstellungen, Datentypen und die Anzahl der zu ändernden Adressen angewiesen werden. Werden keine Schalter verwendet, so gelten die standardmäßig eingestellten Werte.

/P	Physische Adresse im Hauptspeicher (Standard)
/V	Virtuelle Adresse
/CON	Adresse im Speicher des Konsolrechners
/I	Interne Prozessorregister
/G	Allgemeine Register 0...F (R0...PC)
/VB	VBUS-Register
/ID	IDBUS-Register

Bei Verwendung der symbolischen Bezeichnungen können die entsprechenden Schalter entfallen. Für virtuelle Adressen werden die transformierten physischen Adressen verwendet.

/H	Hexadezimaleingabe (Standard nach dem Booten)
/O	Oktaleingabe
/B	Byteeingabe
/W	Worteingabe
/L	Langworteingabe (Standard nach dem Booten)
/Q	Quadworteingabe
/N:x	Änderung von zusätzlich x Adressinhalten.

Bsp.:

(eingestellter Standard sei HEX, LONG, PHYS)

```
>>>DEP/BYT 1234 44<CR>   !Laden der Byteadresse 1234
                         !mit dem Wert 44

>>>D PC 870<CR>          !Laden des PC mit 870

>>>D - 0<CR>             !Laden des SP mit 0

>>>D/N:2 %X200 0<CR>     !Löschen der Adressen 200, 204 und 208

>>>
```

3.5.6. DIRECTORY (DIR)

Format: DIR [<device>]

Funkt.: Es wird das aktuelle Inhaltsverzeichnis der Diskette auf dem angegebenen Laufwerk auf dem Konsolterminal angezeigt. Wird <device> nicht angegeben, wird das Verzeichnis der Konsoldiskette angezeigt.

3.5.7. DISABLE DUAL (DIS DU)

Format: DIS DU

Funkt.: Dieses Kommando verhindert den Zugriff zum zweiten Floppylaufwerk, während der Zugriff zur Konsoldiskette uneingeschränkt möglich ist. Nach Ausführung des Kommandos wird der Zustand der beiden Laufwerke automatisch angezeigt.

- Konsoldiskette auf Laufwerk 0
  DRIVE_0=DX0=CONS[=CSAx], DRIVE_1=DIS oder
  DRIVE_0=DX0=CONS[=CSAx], NO DRIVE_1

  Konsoldiskette auf Laufwerk 1
  DRIVE_0=DIS, DRIVE_1=DX1=CONS[=CSAx] oder
  NO DRIVE_0, DRIVE_1=DX1=CONS[=CSAx]

Dieser Zustand wird nach dem Bootstrap bzw. Rebootstrap des Konsolprogramms automatisch eingenommen und mit dem Kommando 'SHOW' angezeigt. Die Anzeige von 'CSAx' erfolgt erst nach Systemstart.

3.5.8. DUPLICATE (DUP)

Format: DUP

Funkt.: Duplizieren einer vollständigen Diskette.
Als zusätzliche Parameter werden bei Kommandoausführung die Nummern von Ein- und Ausgabelaufwerk angefordert. Es werden RT-11-Disketten vorausgesetzt, die mit Offset=1 formatiert sind. Damit wird eine Duplizierzeit von 90 s erreicht, bei größeren Offsets ist die Zeit geringfügig und bei Offset=0 wesentlich länger.

3.5.9. ENABLE DUAL (EN DUAL)

Format: EN DUAL

Funkt.: Erlauben der Zugriffsmöglichkeit auf das zweite Floppylaufwerk, falls dieses initialisiert ist, d.h. physisch ein System mit 'Dual Drive' vorliegt. Nach Ausführung des Kommandos wird der Zustand der beiden Laufwerke automatisch angezeigt.

- Konsoldiskette auf Laufwerk 0
  DRIVE_0=DX0=CONS, DRIVE_1=DX1[=CSAx] oder
  DRIVE_0=DX0=CONS[=CSAx], NO DRIVE_1

- Konsoldiskette auf Laufwerk 1
  DRIVE_0=DX0[=CSAx], DRIVE_1=DX1=CONS oder
  NO DRIVE_0, DRIVE_1=DX1=CONS[=CSAx]

  Die Anzeige von 'CSAx' erfolgt erst nach Systemstart mit dem 'BOOT'-Kommando.

  Nach dem Bootstrap des RVS K-1840-Betriebssystems wird automatisch der logische Name CSAx: dem Laufwerk zugewiesen, auf dem die Konsoldiskette n i c h t liegt (Schutz der Konsoldiskette vor unbefugtem Zugriff durch Systemnutzer).

Ein Systemzugriff zur Konsoldiskette ist durch folgende Manipulation möglich (Konsoldiskette liegt z.B. auf DRIVE_1):

- Rückgang in den Konsolmodus mit CTRL/P
- Anzeige des automatisch eingestellten Zustandes mit 'SHOW'

DRIVE_0=DX0=CSAx,DRIVE_1=DX1=CONS (A)

- Sperren des Zugriffs zum zweitem Laufwerk durch das Kommando 'DISABLE DUAL' zwingt auch CSAx auf das Konsollaufwerk:

DRIVE_0=DIS,DRIVE1=DX1=CONS=CSAx (B)

- Fortsetzung der Systemarbeit mit dem Kommando 'SET TERMINAL PROGRAMM'

Durch das Kommando 'ENABLE DUAL' kann der zuvor automatisch eingestellte Zustand (A) wiederhergestellt werden.

3.5.10. EXAMINE (E)

Format: E[/sw] [<adr>]

Funkt.: Anzeige des Adressbereiches und des Inhalts der Speicherzelle <adr>. Wird keine Adresse angegeben, so wird der Inhalt der aktuellen Adresse angezeigt; die aktuelle Adresse ergibt sich dabei aus der zuletzt verwendeten Adresse und der dem Datentyp entsprechenden Adresserhöhung:

Byte: +1
Wort: +2
Long: +4
Quad: +8

Sonderregelungen gelten u.a. für die allgemeinen Register R0 bis R15, die generell um 1 weitergestellt Berden. Analog werden die Prozessorregister und die internen Register behandelt.

Es gelten die gleichen Schalter wie beim DEPOSIT-Kommando.

Bsp.:

```
>>>EXAM/PHYS 1234<CR>    !Anzeige der Adresse 1234

        P   00001234   0056FFFF
>>>E/G F<CR>             !Anzeige des PC

        G   0000000F   00000865
>>>E SP<CR>              !Anzeige des SP

        G   0000000E   8F00FE84
>>>E +<CR>               !Anzeige des PC

        G   0000000F   00000865
>>>E @<CR>               !Anzeige der nächsten Befehlsfolge

        P   00000865   A05354D0
>>>E/CON/WO %O20000<CR> !Anzeige der Konsoladresse 20000
```

```
        C   00020000   000000
>>>E<CR>                !Anzeige der Folgeadresse

        C   00020002   012345
>>>
```

Format: E IR

Funkt.: Anzeige des Befehls-Registers.

Bsp.:

```
>>>E IR

        IR 90  00  01
>>>
```

3.5.11. FORMAT (FO)

Format: FO

Funkt.: Formatieren einer Diskette. Als zusätzliche Parameter werden bei der Kommandoausführung angefordert:

Offset: Parameter zur Bestimmung der Reihenfolge der Sektoren innerhalb einer Spur.
Es sind zu verwenden:
Für DS0 (LPROEM): Offset = A
Für DS1 ... DSx (LPROEM): Offset = 1

TO Drive: Physische Laufwerknummer

Das Ende des eigentlichen Formatierens wird angezeigt. Die nachfolgende Prüfung der Diskette durch Schreiben und Kontroll-Lesen eines Testmusters kann ggf. mit CTRL/C abgebrochen werden.

3.5.12. HALT (H)

Format: H

Funkt.: Das HALT-Kommando veranlasst ein Anhalten der CPU nach Beendigung der Abarbeitung der laufenden Befehls. Es erfolgt die Meldung

```
        HALTED AT xxxxxxxx
>>>
```

auf dem Konsolterminal; xxxxxxxx ist dabei der Wert der momentanen Programmzählerstandes.

3.5.13. HELP (HE)

Format: HE [<filespec>]

Funkt.: Mit dem Kommando HELP werden stets die entsprechenden Dateien auf der Konsoldiskette angesprochen. Wird das Kommando HELP ohne Angabe eines Files verwendet, erfolgt eine Ausgabe des Files CONSOL.HLP auf dem

Konsolterminal (äquivalent zu HE CONSOL). Fuer die anderen Helpdateien ist dann entsprechend

HE ABBREV[.HLP]	fuer die Abkürzungen
HE ERROR[.HLP]	fuer die Fehlermeldungen
HE BOOT[.HLP]	fuer Informationen zum Booten

einzugeben. Dabei kann der Dateityp .HLP entfallen.

3.5.14. INITIALIZE (I)

Format: I

Funkt.: Dieses Kommando veranlaßt eine Initialisierung der CPU. Danach erfolgt die Ausschrift

```
        INIT SEQ DONE
>>>
```

3.5.15. LINK (LI)

Format: LI

Funkt.: Eröffnen eines Indirektkommandofiles. Das Konsolprogramm zeigt mit der Ausschrift "<<<" den LINK-Modus an. Alle nachfolgend eingegebenen Kommandos werden in einem Indirektkommandofile abgespeichert. CONTROL/C (^C) beendet das LINK-Kommando (siehe auch PERFORM). Das Indirektfile kann eine Größe von maximal 10 Sektoren nicht überschreiten. Dieses File wird auf feste Sektoren der Konsoldiskette geschrieben und nicht im Inhaltsverzeichnis der Diskette eingetragen!

3.5.16. LOAD (L)

Format: L <filspec> oder
L/S:<adr> <filespec>

Funkt.: Laden des angegebenen Files in den Hauptspeicher ab Adresse 0 bzw. ab Adresse <adr>.

Format: L/WC[S] <filespec>

Funkt.: Laden des spezifizierten Files in den WCS.

3.5.17. NEXT (N)

Format: N <zahl>

Funkt.: Ausführung von <zahl> Schritten, die Schrittart hängt vom letzten SET STEP-Kommando ab.

3.5.18. OVERLAY (O)

Format: O <filespec>

Funkt.: Das Kommando OVERLAY wird verwendet, um nutzerspezifische Programme in den RAM-Bereich des Konsolprogramms zu laden. Dabei wird das Konsolprogramm überschrieben, jedoch nicht die Interruptvektoren im Bereich bis Adresse %0776. Bei Beendigung dieses Programms ist ein Neuladen des Konsolprogramms zu veranlassen (mit JMP @#140xxx). Die Filespezifikation kann dabei eine Geräteangabe DX0: oder DX1: enthalten.

3.5.19. PERFORM (P)

Format: P

Funkt.: Abarbeiten eines mit LINK erstellten Indirektkommandofiles. Ist das Kommando PERFORM selbst Bestandteil des Files (dann nur an letzter Position sinnvoll), so erfolgt eine ununterbrochene Abarbeitung dieses Files. Dies ist z.B. sinnvoll, wenn Geräte-Register oder andere Speicherzellen auf Änderung von bestimmten Bitpositionen getestet werden sollen.

3.5.20. QCLEAR (Q)

Format: Q <adr>

Funkt.: Ausführung eines Quad Clear mit der Adresse <adr>, die auf eine Quadwortgrenze gebracht wird, indem die niederwertigsten drei Adressbits gelöscht werden. Die Adresse <adr> wird generell als physische Hauptspeicheradresse interpretiert. Sie dient dem Löschen von nichtkorrigierbaren ECC-Fehlern.

3.5.21. REBOOT (REB)

Format: REB

Funkt.: Das REBOOT-Kommando hat ein Neuladen des Konsolprogramms zur Folge ohne Beeinflussung der CPU. Es erfolgt kein Neuladen des WCS.

3.5.22. REPEAT (R)

Format: R <konsolkommando>

Funkt.: Abarbeitung des Konsolkommandos <konsolkommando> ununterbrochen bis zur Eingabe von CONTROL/C (^C). Dabei kann jedes Kommando außer REPEAT verwendet werden.

3.5.23. SET (SE)

Format: SE D[EFAULT] [<option>,...]

Funkt.: Setzen von Standardwerten im Konsolprogramm. Werden keine Werte für <option> eingegeben, werden als Standard wie beim Start des Konsolprogramms H[EX], L[ONG] und P[HYS] eingestellt. Die möglichen Werte von <option> sind:

O[CTA]
H[EX]
P[HYS]
V[IRT]
I[NTERN]
CON[S]
G[ENE]
VB[US]
ID[BU]
B[YTE]
W[ORD]
L[ONG]
Q[UA]

Format: SE S[TEP] B[US]

Funkt.: Einschalten des Einzelbuszyklustakt-Modus.

Format: SE S[TEP] S[TATE]

Funkt.: Einschalten des Einzelzeittakt-Modus.

Format: SE S[TEP] I[NSTRUCTION]

Funkt.: Einschalten des Einzelbefehls-Modus.

Beim Start des Konsolprogramms ist standardmäßig kein Schrittbetrieb eingestellt. Wird das 'SET STEP'-Kommando ohne Zusatzangabe verwendet, so wird der Schrittbetrieb ausgeschaltet (gleiche Wirkung wie CLEAR STEP).

Format: SE T[ERMINAL] F[ILL]:<zahl>

Funkt.: Setzen des Zählers für die Anzahl der Leerzeichen, die nach <CR> oder <LF> auf das Terminal geschrieben werden.

Format: SE T[ERMINAL] PR[OGRAM]

Funkt.: Umschalten des Konsolterminals auf Programm-E/A-Modus.

Format: SE SO[MM]

Funkt.: Setzen der 'STOP ON MICROBREAK MATCH' (SOMM)-Erlaubnis auf der CIB. (Micromatch-Register: ID-Register %X21). Ist SOMM gesetzt und der Inhalt des K 1840-Mikro-PC ist identisch mit dem Inhalt des ID-Registers %X21, so wird der CPU-Takt angehalten.

Beim Start des Konsolprogramms ist standardmäßig CLEAR SOMM eingestellt.

Format: SE C[LOCK] S[LOW]

Funkt.: Setzen der CPU-Taktfrequenz auf 'LANGSAM' (8.925 MHz).

Format: SE C[LOCK] F[AST]

Funkt.: Setzen der CPU-Taktfrequenz auf 'SCHNELL' (10.525 MHz).

Format: SE C[LOCK] N[ORMAL]

Funkt.: Setzen der CPU-Taktfrequenz auf 'NORMAL' (10.0 MHz).

Beim Start des Konsolprogramms ist die Taktfrequenz standardmäßig auf 'NORMAL' eingestellt.

Format: SE R[ELOCATION]:<zahl>

Funkt.: Eingabe von <zahl> auf das Verschieberegister des Konsolprogramms (32-Bit-Wert).
Der Wert dieses Registers wird generell zur effektiven Adresse von physischen und virtuellen Speicherzugriffen in den Kommandos EXAMINE und DEPOSIT addiert.

Beim Start des Konsolprogramms hat das Verschieberegister den Wert 0.

3.5.24. SHOW (SH)

Format: SH

Funkt.: Anzeige des Konsol- und CPU-Status, z.B.:

```
>>>SH

        CPU HALTED,SOMM CLEAR,STEP=NONE,CLOCK=NORM
        RAD=OCT,ADD=CONS,DAT=WORD,FILL=000,REL=00000000
        (Zustand der Laufwerke 0 und 1 siehe Abschnitt 3.3.)
>>>
```

Format: SH V[ERSION]

Funkt.: Anzeige der Versionen von Konsole und Mikroprogramm, z.B.:

```
>>>SH V

        VER: PCS=01 WCS=0E-10 FPLA=0E CON=V01-00-L
>>>
```

3.5.25. START (S)

Format: S <adr>

Funkt.: Das S <adr>-Kommando dient zum Starten eines Programms im Hauptspeicher der CPU und ist mit der folgenden Kommandofolge identisch:

```
>>>I                    !Initialisieren der CPU

>>>D PC <adr>           !Eintragen der Startadresse in den PC

>>>C                    !Start der Befehlsabarbeitung in der CPU.
```

Danach befindet sich das Konsolterminal im Programm-E/A-Modus.

Format: S/WC[S] <adr>

Funkt.: Das S/WCS <adr>-Kommando dient dem Starten eines Mikroprogramms und ist mit der Kommandofolge

```
>>>D/I 2C <adr>         !Eintragen von <adr> in den Micro-PC

>>>CLEAR STEP           !Starten des Taktes
```

gleichwertig.

3.5.26. TEST (T)

Format: T [<device>]

Funkt.: Laden und Starten der Mikrodiagnose (MICMON.SYS und untergeordnete Monitore). Dabei werden alle Teile der Mikrodiagnose abgearbeitet. Mit der Geräteangabe <device> kann ausgewählt werden, von welchem Laufwerk die Mikrodiagnose gestartet werden soll. Wird <device> nicht angegeben, wird der Mikrodiagnose-Monitor auf dem Laufwerk gesucht, auf dem die Konsoldiskette liegt.

Format: T/C[OM] [<device>]

Funkt.: Laden der Mikrodiagnose und Aufruf des Kommando-Interpreters. Nach der Meldung des Monitors mit

```
MIC>
```

ist eine gezielte Auswahl von Testsektionen bzw. Tests, die abgearbeitet werden sollen, möglich. Weitere Details sind den Beschreibungen zur Mikrodiagnose zu entnehmen.

3.5.27. TYPE (TY)

Format: TY <filespec>

Funkt.: Ausgabe eines beliebigen Disketten-Textfiles über Terminal.

3.5.28. UNJAM (U)

Format: U

Funkt.: Rücksetzen des Internen Synchronbusses (ISB).

3.5.29. WAIT DONE (WA D)

Format: WA D

Funkt.: Im Falle der Verarbeitung von einem Indirektkommandofile aus wird die Abarbeitung dieses Files unterbrochen bis:

- ein 'DONE'-Signal von einem Programm in der CPU empfangen wird (—> die Verarbeitung des Files wird fortgesetzt) oder
- die CPU geht in den HALT-Zustand bzw. der Operator gibt CONTROL/C (^C) ein (—> die Verarbeitung des Files wird beendet)

3.5.30. WCS (W)

Format: W [<device>]

Funkt.: Aufruf des Mikro-Debuggers.

3.5.31. Indirekt-Kommando (@)

Format: @<filespec>

Funkt.: Eröffnen des Indirektkommandofiles mit der File-Bezeichnung <filename> und Beginn der Abarbeitung der im File enthaltenen Konsolkommandos. Die Verarbeitung wird solange fortgesetzt, bis eine der folgenden Bedingungen erfüllt ist:

- ein WAIT DONE-Kommando wird vom File gelesen
- das Ende der Datei wird erreicht; dabei wird <@EOF> ausgegeben und eine Kommandoanforderung auf dem Konsolterminal ausgegeben
- ein CTRL/C wird vom Konsolterminal eingegeben; in diesem Fall wird die Bearbeitung der Kommandodatei abgebrochen.

3.5.32. CONTROL/P (^P)

Format: ^P

Funkt.: Umschalten des Konsolterminals auf Konsol-E/A-Modus (Übergang in den Konsoleingabemodus).

3.6. Bedingungen für die Kommandoverarbeitung

Die meisten Konsolkommandos erfordern, daß sich die CPU im HALT-Zustand befindet. Einige Kommandos, die keine Zusammenarbeit mit der CPU erfordern, können auch bei arbeitender CPU ausgeführt werden. Diese Kommandos sind:

- SHOW
- HELP
- SET-Kommandos
- EXAMINE VBUS
- HALT
- CLEAR
- WAIT DONE

Werden andere Konsolkommandos als die soeben genannten eingegeben, während die CPU arbeitet, so erfolgt ein Abbruch der Kommandobearbeitung und die Fehlermitteilung

```
        ?CPU NOT IN CONSOLE WAIT LOOP,FUNCTION ABORTED
>>>
```

wird auf das Konsolterminal ausgegeben.

3.7. Kommentare innerhalb von Kommandos

Die Kommandoverarbeitung des Konsolprogramms erlaubt die Verwendung von Kommentaren innerhalb und außerhalb von Konsolkommandos. Kommentare beginnen mit einem Ausrufezeichen. Wird in einer Eingabezeichenfolge ein Ausrufezeichen erkannt, so wird die gesamte restliche Zeile als Kommentar angesehen und die Kommandoanalyse beendet. Damit können auch Indirektkommandofiles Kommentare enthalten, da sie nur eine Folge von Konsolkommandos darstellen. Insbesondere enthalten alle Help-Files ausschließlich Kommentarzeilen.

Bsp.:

```
>>>!BEISPIEL FUER EIN EXAMINE-KOMMANDO

>>>E/G F        !CPU-REGISTER 15 = PC

        G  0000000F  00001234
>>>
```

3.8. Helpfile zum Konsolprogramm

```
!Revision 1.1
!Fuer Systeme mit 5 1/4"-Floppylaufwerken
!
!
!                    Helpfile zum Konsolprogramm
!
!
!Weitere Helpfiles:
!Fuer Abkuerzungsregeln        : 'HE ABBREV'  eingeben.
!Fuer Fehlernachrichten        : 'HE ERROR' eingeben.
!Fuer BOOT-Parameter    :        'HE BOOT' eingeben
!
!Fuer Ausgabestop                'CONTROL/C' eingeben
!
!
!
!Syntax der Konsolkommandosprache:
!        Alle Kommandos werden mit <CR> abgeschlossen.
!
!       'EXAMINE' und 'DEPOSIT' Schalter fuer den Adressraum :
!               '/P'          = physischer Speicher (Standard)
!               '/V'          = virtueller Speicher
!               '/I'          = interne Register (Prozessor)
!               '/G'          = Register 0 bis F (R0 bis PC)
!               '/VB'         = VBUS-Register
!               '/ID'         = IDBUS-Register
!               '/CONSOLE     = Adresse im Speicher des Konsolrechners
!
!       'EXAMINE' und 'DEPOSIT' Schalter fuer Datenlaengen :
!               '/B'  = Byte (8 Bits)
!               '/W'  = Wort (2 Bytes)
!               '/L'  = Langwort (2 Worte)
!               '/Q'  = Quadwort (4 Worte)
!
!       <adr> ist eine <zahl>, oder eine der folgenden symbolischen
!       Adressen:
!               'R0,R1,R2,......,R11,AP,FP,SP,PC'     (allgemeine Register)
!               'PSL' = Prozessorstatuslangwort
!               '*'   = zuletzt benutzte Adresse
!               '+'   = der letzten Adresse (*) nachfolgende Adresse
!               '-'   = der letzten Adresse (*) vorhergehende Adresse
!               '@'   = verwendet letzten EXAMINE/DEPOSIT-Datenwert als Adresse
!
!       <zahl>  =  Zeichenfolge im augenblicklich eingestelltem
!                  Zahlensystem oder durch vorangestelltes %O als Oktalzahl
!                  bzw. durch %X als Hexadezimalzahl gekennzeichnet.
!
!       <filespec>=<dev>:<filename>
!                  Spezifikation von Files auf dem Konsolfolienspeicher.
!                  <dev>: logischer Geraetename der Laufwerke (-/DX0/DX1).
!                  Die Zuordnung von logischen zu physischen Geräetenamen
!                  (DRIVE 0/1) und deren Anzeige erfolgt mit den Kommandos
!                  'ENABLE DUAL' und 'DISABLE DUAL' ('DUAL' von 'DUAL DRIVE'
!                  =Arbeit mit zwei Laufwerken).
!                  Ausserdem erfolgt eine Anzeige mit dem 'SHOW'-Kommando.
```

```
!                   Wird <dev>: nicht angegeben, erfolgt stets Zugriff zur
!                   Konsoldiskette, unabhaengig davon, auf welchem Laufwerk
!                   diese liegt.
!                   'DX0:' oder 'DX1:' koennen nur angegeben werden, wenn
!                   sie einem physischen Laufwerk zugewiesen sind.
!                   Mit dem Kommando 'BOOT' wird im Falle des Vorhandenseins
!                   von zwei Laufwerken der Zugriff zum zweiten Laufwerk er-
!                   laubt. Der Systemzugriff ueber 'CSA0" im Diagnosesuopervi-
!                   sor bzw. 'CSA1' im SCP wird dann stets auf das Laufwerk
!                   gelenkt, auf dem die Konsoldiskette n i c h t liegt
!                   (Schutz derselben vor unbefugtem Zugriff durch System-
!                   nutzer).
!                   'CSAx' kann auf die Konsoldiskette gelenkt werden, indem
!                   mit 'CONTROL/P' in den Konsolmodus gegangen wird, dort
!                   'DISABLE DUAL' gegeben wird und mit 'SET TERMINAL PROGRAM"
!                   zurückgegangen wird.
!
!
!'BOOT'                    -Booten der CPU vom Standardgeraet ueber das Indi-
!                           rekt-Kommandofile DEFBOO.CMD auf der Konsoldiskette.
!
!'BOOT [<dev>:] <bootdev>'-Durch <bootdev> wird ein Geraet bzw. ein auf
!                          dieses zugeschnittenes Indirekt-Kommandofile fuer
!                          das Booten definiert.
!                          <bootdev> wird zur Bildung des Filenamens benutzt
!                          und muss deshalb 3 Alphazeichen lang sein.
!                          Die Diskette mit diesem Kommandofile kann auf einem
!                          beliebigen Laufwerk liegen (durch <dev> definiert).
!                          Mit dem Kommando 'BOOT DX1:ABC' wird z.B. das
!                          Komandofile ABCBOO.CMD von Laufwerk DX1 gerufen.
!
!'CLEAR STEP'              -Aufheben jedes eingestellten Taktschritt-
!                           Modus, Uebergang in den Normalmodus.
!
!'CLEAR SOMM'              -Aufheben der 'STOP ON MICROBREAK MATCH'
!                           (SOMM)-Erlaubnis.
!                           Merke: ID-Register 21 ist das
!                           MIKRO-MATCH -Register.
!
!'CONTINUE'                -Programmfortsetzung.
!
!'DEPOSIT[/<schalter>] <adr> <dat>'    -Abspeichern der <dat> auf die
!                                       Speicherzelle <adr>.
!
!'DIRECTORY [<dev>:]       -Anzeige des Inhaltverzeichnisses
!                           der Diskette auf dem mit
!                           <dev>: angesprochenen Geraet.
!
!'DUPLICATE'               -Duplizieren einer vollstaendigen Diskette.
!                           Als zusaetzliche Parameter werden die physischen
!                           Nummern der Ein- und Ausgabelaufwerke angefordert.
!                           Es werden RT-11-Disketten vorausgesetzt, die mit
!                           OFFSET=1 formatiert sind. Damit wird eine Dupli-
!                           zierzeit von 90 s erreicht, bei groesseren Offsets
!                           ist die  Zeit etwas und bei Offset=0 wesentlich
!                           laenger.
!
!'DISABLE DUAL'            -Sperren des Zugriffs zum zweiten  Floppylaufwerk.
```

```
!                         Dieser Zustand wird nach dem Booten des Konsolpro-
!                         grammes automatisch eingestellt.
!
!'ENABLE DUAL'           -Freigabe des Zugriffs zum zweiten Floppylauferk.
!                         Dieser Zustand wird  bei Ausfuehrung des Kommandos
!                         'BOOT' automatisch eingestellt, falls zwei Lauf-
!                         werke physisch vorhanden sind ('DUAL DRIVE').
!
!'EXAMINE[/<schalter>] <adr>'          -Anzeige des Speicherinhaltes ent-
!                                       sprechend <adr> und gesetzten
!                                       Schaltern.
!
!'EXAMINE IR'            -Anzeige des Befehlsregisters IR.
!
!'FORMAT'                -Formatieren einer Diskette.
!                         Als zusaetzliche Parameter werden angefordert:
!                         OFFSET:   Parameter zur Bestimmung der Reihen-
!                                   folge der Sektoren innerhalb einer Spur.
!                                   Es sind zu verwenden:
!                                   Fuer DS0       (LPROEM):     OFFSET=A
!                                   Fuer DS1...DSx (RT-11 ):     OFFSET=1
!                         TO DRIVE: Physische Laufwerksnummer.
!                         Das Ende des eigentlichen Formatierens wird ange-
!                         zeigt. Das nachfolgende Kontrollschreiben und
!                         -lesen kann ggf. mit CONTROL/C abgebrochen werden.
!
!'HALT'                  -Anhalten der CPU.
!
!'HELP [<filename>]'     -Aufforderung zur Ausgabe des
!                         gewuenschten Files.
!
!'INITIALIZE'            -Initialisieren der CPU.
!
!'LINK'                  -Eroeffnen eines Indirektkommandofiles.
!                         Das Konsolprogramm zeigt mit "<<<" den
!                         den Link-Modus an.
!                         Alle Kommandos die waehrend des Linkens
!                         vom Nutzer eingegeben werden, werden in
!                         einem indirekten Kommandofile fuer
!                         eine spaetere Abarbeitung gespeichert.
!                         Control/C beendet das Linken.
!                         (siehe PERFORM)
!
!'LOAD[/START:<adr>] <filespec>'       -Laden des Files in den Hauptspeicher,
!                                       Starten ab Adresse 0, oder wenn
!                                       spezifiziert ab <adr>.
!
!'LOAD/WCS <filespec>'                 -Laedt spezifizierten File in den WCS.
!
!'NEXT <zahl>'           -<zahl> Stepzyklen wurden ausgeführt.
!                         Step-Typ hängt von dem letzten
!                         'SET STEP'-Kommando ab.
!
!'OVERLAY <filespec>'    -Laden des durch <filespec> definierten Programms
!                         in den K 1620 (ueber das CONSOL.SYS) und Start
!                         dieses Programmes.
!
!'PERFORM'               -Abarbeiten eines mit LINK erstellten
```

```
!                          Indirektkommandofiles.
!
!'QCLEAR <adr>'           -Fuehrt ein Quad Clear mit der
!                          <adr> aus. Dies ist sinnvoll fuer
!                          eine Quadwortgrenze.
!                          (Loescht ECC Errors.)
!
!'REBOOT'                 -Rueckladen der Konsolsoftware.
!                          Ein Umlegen der Konsoldiskette auf ein anderes
!                          Laufwerk ist dabei zulaessig.
!
!'REPEAT <konsolkommando>'-Wiederholung der Abarbeitung des
!                          <konsolkommando>, bis Control/C (^C)
!                          eingegeben wird.
!
!'SET CLOCK SLOW'         -Setzen der CPU-Taktrate auf "Langsam"
!                          (8.925 MHz).
!
!'SET CLOCK FAST'         -Setzen der CPU-Taktrate auf "Schnell"
!                          (10.525 MHz).
!
!'SET CLOCK NORMAL'       -Setzen der CPU-Taktrate auf "Normal"
!                          (10.0 MHz).
!
!'SET DEFAULT [<option>,...,<option>]' -Setzen des Konsolstandardes.
!                                       Merke: <option> sind: OCTAL, HEX,
!                                       PHYSICAL, VIRTUAL, INTERNAL, GENERAL,
!                                       VBUS, IDBUS, BYTE, WORD, LONG und
!                                       QUAD.
!
!'SET RELOCATION:<zahl>' -Eingabe der <zahl> auf das Verschie-
!                          bungsregister des Konsolprogramms.
!                          Der Wert dieses Registers wird generell
!                          zur effektiven Adresse von physischen
!                          und virtuellen Speicherzugriffen in den
!                          Kommandos 'EXAMINE' und 'DEPOSIT" addiert.
!
!'SET SOMM'               -Setzen der 'STOP ON MICROBREAK MATCH'
!                          (SOMM)-Erlaubnis auf der CIB.
!
!'SET STEP BUS'           -Setzen des Einzelbuszyklustakt-Modus.
!
!'SET STEP INSTRUCTION'   -Setzen des Einzelbefehls-Modus.
!
!'SET STEP STATE'         -Setzen des Einzelzeittakt-Modus.
!
!'SET TERMINAL FILL:<zahl>'   -Setzen des Zaehlers fuer die Anzahl
!                              der Leerstellen, die nach <CR> o. <LF>
!                              auf das Terminal geschrieben werden.
!
!'SET TERMINAL PROGRAMM' -Umschalten des Konsolterminals
!                          auf Programm-E/A-Modus.
!
!'SHOW'                   -Anzeige des Status von Konsole und CPU
!                          (CPU, SOMM, STEP, RAD, ADD, DAT,
!                          FILL, REL).
!
!'SHOW VERSION'           -Anzeige der Versionen des Mikro-
```

```
!                          programmes und der Konsole
!                          (PCS, WCS, FPLA, CON).
!
!'START <adr>'            -Initialisiert CPU, lagert <adr>
!                          im PC ab, Continue, Programmstart.
!
!
!'TEST [<dev>:]'          -Abarbeitung der Mikrodiagnoseprogramme (Laden vom
!                          durch <dev>: spezifizierten Laufwerk, Start).
!                          Wird <dev>: nicht angegegeben, ist die Mikrodiagnose-
!                          diskette an Stelle der Konsoldiskette zu legen.
!
!'TEST/COM [<dev>:]'      -Laden des Mikrodiagnose-Monitors vom durch <dev>:
!                          spezifizierten Laufwerk mit Warten auf Kommandos.
!
!'TYPE <filespec>'        -Ausgabe eines beliebigen Textfiles ueber Terminal.
!
!'UNJAM'                  -Ruecksetzen des Internen Synchronbusses.
!
!'WAIT DONE'              -Erfolgt die Verarbeitung von einem
!                          indirekten Kommandofile, wird dieses
!                          Kommando die Abarbeitung des
!                          Kommandofiles stoppen bis:
!                                 A)  Ein 'DONE' Signal von dem
!                                     laufenden Programm in der CPU
!                                     empfangen wurde. (Kommandoab-
!                                     abarbeitung wird weitergefuehrt).
!                          oder:  B)  Die CPU geht in den Halt-
!                                     Zustand ueber o. der
!                                     Operator gibt ein Control-C
!                                     ein. (^C: Abarbeitung des Kom-
!                                     mandofiles wird beendet).
!
!'WCS [<dev>:]            -Aufruf des Mikrodebuggers vom durch <dev>: spezi-
!                          fizierten Floppylaufwerk.
!                          Wird <dev>: nicht angegeben, wird die Mikrodebug-
!                          gerdiskette an Stelle der Konsoldiskette
!                          erwartet.
!
!'^P'(CONTROL/P)          -Uebergang des Konsolterminals in den
!                          Konsol-E/A-Modus,
!                          (sofern der Modusschalter sich nicht
!                          im 'DISABLE' Zustand befindet).
!
!'@<filespec>'            -Eroeffnen des Indirektkommandofiles
!                          mit der Filebezeichnung <filespec>
!                          und Beginn der Abarbeitung der in dem
!                          File enthaltenen Konsolkommandos.
!
!<Ende-von-CONSOL.HLP>
```

3.9. Helpfile für Abkürzungen der Konsolkommandosprache

```
!Revision 1.1
!Fuer Systeme mit 5 1/4"-Floppylaufwerken
!
!       Helpfile fuer Abkuerzungen der Konsolkommandosprache
!       ----------------------------------------------------
!
! Diese Datei zeigt die Abkuerzungen der einzelnen Kommandos und der
! Schalter.
!
!          Kommando                    kuerzeste Kommandobezeichnung
!
!  'BOOT'                                  'B'
!  'CLEAR SOMM'                            'CL SO'
!  'CLEAR STEP'                            'CL S'
!  'CONTINUE'                              'C'
!  'DEPOSIT <adr> <dat>'                   'D <adr> <dat>'
!  'DIRECTORY <dev>'                       'DIR <dev>'
!  'DISABLE DUAL'                          'DIS DU'
!  'DUPLICATE"                             'DUP'
!  'ENABLE DUAL'                           'EN DU'
!  'EXAMINE <adr>'                         'E <adr>'
!  'EXAMINE IR'                            'E IR'
!  'FORMAT'                                'FO'
!  'HALT'                                  'H'
!  'HELP'                                  'HE'
!  'INITIALIZE'                            'I'
!  'LINK'                                  'LI'
!  'LOAD <filespec>'                       'L <filespec>'
!  'NEXT <zahl>'                           'N <zahl>'
!  'OVERLAY <filespec>'                    'O <filespec>'
!  'PERFORM'                               'P'
!  'QCLEAR <adr>'                          'Q <adr>'
!  'REBOOT'                                'REB'
!  'REPEAT <konsolkommando>'               'R <konsolkommando>'
!  'SET CLOCK FAST'                        'SE C F'
!  'SET CLOCK SLOW'                        'SE C S'
!  'SET CLOCK NORMAL'                      'SE C N'
!  'SET RELOCATION:<zahl>'                 'SE R:<zahl>'
!  'SET SOMM'                              'SE SO'
!  'SET STEP BUS'                          'SE S B'
!  'SET STEP STATE'                        'SE S S'
!  'SET STEP INSTRUCTION'                  'SE S I'
!  'SET TERMINAL FILL:<zahl>'              'SE T F:<zahl>'
!  'SET TERMINAL PROGRAM'                  'SE T PR'
!  'SET DEFAULT <option>'                  'SE D <option>'
!  'SHOW'                                  'SH'
!  'SHOW VERSION'                          'SH V'
!  'START <adr>'                           'S <adr>'
!  'TEST'                                  'T'
!  'TYPE'                                  'TY'
!  'UNJAM'                                 'U'
!  'WAIT DONE'                             'WA D'
!  'WCS[<dev>]'                            'WC<dev>'
!  '@<filespec>'                           '@<filespec>'
```

```
!
!         Schalter                    kuerzeste Scvhalterbezeichnung
!
!  /CONSOLE                                  /CON
!  /BYTE                                     /B
!  /WORD                                     /W
!  /LONG                                     /L
!  /QUAD                                     /Q
!  /OCTAL                                    /O
!  /HEX                                      /H
!  /PHYSICAL                                 /P
!  /VIRTUAL                                  /V
!  /INTERNAL                                 /I
!  /GENERAL                                  /G
!  /VBUS                                     /VB
!  /IDBUS                                    /ID
!  /WCS                                      /WC
!  /NEXT:<zahl>                              /N:<zahl>
!  /COMMAND                                  /C
!  /START:<adr>                              /S:<adr>
!
!<Ende-von-ABBREV.HLP>
```

3.10. Bootstrap-Helpfile für VMB.EXE

```
!Revision 1.1
!
!       Bootstrap-Helpfile fuer VMB.EXE
!       -------------------------------
!
!       Diese Datei beschreibt die Eingabeparameter fuer das Bootstap-
!       Programm VMB.EXE. Im Normalfall wird der Bootstrap-Prozess den
!       File [SYSEXE]SYSBOOT.EXE auf dem spezifizierten Geraet suchen,
!       laedt ihn in den Speicher und uebertraegt die Steuerung zu ihm.
!
!       Zwei Sätze der Kommandofiles werden auf dem RVS/SVP-Konsolfloppy
!       fuer die Ausfuehrung der notwendigen Bootstrap-Operationen
!       geliefert.
!       Ein Satz von diesen Kommandofiles wird die Auswahl einer Option
!       zum Stop im SYSBOOT laden, damit ein Systemparameterwechsel erfolgen
!       kann. Sie werden als indirekte Kommandofiles aufgerufen.
!
!               @DM0GEN                 ! Laden von PK07 Geraet 0
!               @DM1GEN                 !       Geraet 1
!               @DM2GEN                 !       Geraet 2
!               @DM3GEN                 !       Geraet 3
!               @DB0GEN                 ! Laden von PM03/PP06 Geraet 0
!
!               @DB1GEN                 !       Geraet 1
!               @DB2GEN                 !       Geraet 2
!               @DB3GEN                 !       Geraet 3
!               @DB4GEN                 !       Geraet 4
!               @DB5GEN                 !       Geraet 5
!               @DB6GEN                 !       Geraet 6
!               @DB7GEN                 !       Geraet 7
!
!       Der andere Satz wird im Normalfall nur ueber das BOOT-Kommando
!       aufgerufen, kann aber auch explizit als indirektes
!       Kommandofile aufgerufen werden. Diese Kommandofiles
!       fuehren ein normales, nichtwechselwirkendes Laden aus, ohne
!       irgendeinem Stop im SYSBOOT fuer den Parameterwechsel.
!
!       BOOT DM0    oder @DM0BOO.CMD    ! Laden PK07 Geraet 0
!       BOOT DM1                        !       Geraet 1
!       BOOT DM2                        !       Geraet 2
!       BOOT DM3                        !       Geraet 3
!       BOOT DB0                        ! Laden PM03 oder PP06 Geraet 0
!       BOOT DB1                        !       Geraet 1
!       BOOT DB2                        !       Geraet 2
!       BOOT DB3                        !       Geraet 3
!       BOOT DB4                        !       Geraet 4
!       BOOT DB5                        !       Geraet 5
!       BOOT DB6                        !       Geraet 6
!       BOOT DB7                        !       Geraet 7
!
!       Der Bootstrap-Prozess ist im Speicher mindestens eine Page ueber dem
!       ersten verfuegbaren Arbeitsspeicher geladen, damit der Zwischenraum
!       fuer den Restartparameterblock ermoeglicht werden kann. Die Basis-
!       adresse des Bootstrap-Prozesses laeuft uber den Stack-Pointer (SP),
!       welcher auch als temporaerer Stack-Pointer dient.
```

```
!
!  Eingabeparameter:
!
!        R0  -  Bootstrap-Prozess-Geraetetypkode (RPB$B_DEVTYPE)
!               <07:00>
!                        0 => MSBUS-Geraete      (PM03/PM05/PP04/PP05/PP06/
!                                                 PM40
!                        1 => Plattenkassette    (PK06/PK07)
!                        2 => PL01/2
!                        3 => reserviert
!                     4-16 => reserviert fuer SKRBUS-Geraete
!                       17 => SPA50
!                    18-31 => reserviert
!                       32 => HI
!                    33-63 => reserviert fuer SKRBUS-Geraete
!                       64 => Konsolespeicherblockeinheit
!               <15:08> reserviert fuer eine zukuenftige Erweiterung
!               <31:16> Geraeteklassenabhaengigkeit (RPB$W_ROUBVEC)
!                       SKRBUS  - Vektoradresse wahlweise, 0 schliesst die
!                                 Verwendung des Standardvektors ein
!                       MSBUS   - muss Null sein
!
!        R1  -  Bootstrap-Prozess-Geraetebusadresse
!               K 1840
!                        <31:04> muss Null sein
!                        <03:00> TR-Nummer des Adapters
!
!        R2  -  Bootstrap-Prozess-Controller Information:
!               SKRBUS:
!                        <31:18> muss Null sein
!                        <17:00> SKRBUS-Adresse des Geraetesteuerregisters
!
!               MSBUS:
!                        <31:04> muss Null sein
!                        <03:00> Controller/Formatierer-Nummer des Adapters
!
!               CI:
!                        <31:08> muss Null sein
!                        <07:00> HSC-Kanalnummer (Adresse feststehend)
!
!        R3  -  Laden der Geraetenummer
!                        <31:04> muss Null sein
!                        <03:00> Geraetenummer
!
!        R4  -  logische Blocknummer des Boot-Blockes
!
!        R5  -  Software Boot-Steuerflags
!
!                Bit     Erlaeuterung
!                ---     ------------
!                 0      Dialogfaehiges Laden. An verschiedenen Stellen werden
!                        im System des Boot-Verfahrens Parameter und andere Ein-
!                        gaben von der Konsole erbeten.
!
!                 1      Debug. Wenn dieses Flag gesetzt ist, wird SVP den Code
!                        des XDELTA Debugger in die System-Pagetabelle des
!                        laufenden Systemes einbinden.
!
```

```
!                2      Anfangsunterbrechungspunkt. Wenn dieses Flag gesetzt
!                       ist und der XDELTA-Debuggercode wurde eingebunden
!                       (Flagbit 1), dann wird ein Unterbrechungspunkt unmit-
!                       telbar nach dem EXEC freigegebenen Mapping auftreten.
!
!                3      Boot-Block. Wenn dieses Flag gesetzt ist, erfolgt
!                       ein sekundaeres Laden vom Boot-Block. Der sekundaere
!                       Bootstrap-Prozess ist ein einzelner 512 byte Block,
!                       dessen logische Blocknummer in R4 steht.
!
!                4      Diagnose-Laden.  Dieses Flag verursacht ein Laden
!                       des Filenamens fuer den Diagnose-Supervisor
!                       ([SYSMAINT]DIAGBOOT.EXE).
!
!                5      Bootstrap-Unterbrechungspunkt. Dieses Flag veran-
!                       lasst den primaeren und sekundaeren  Bootstrap-Prozess
!                       an einem Unterbrechungspunkt, nach der Ausfuehrung
!                       notwendiger Initialisierungen, zu stoppen.
!
!                6      Image-Kopf. Wenn dieses Flag gesetzt ist, wird die
!                       Uebertragungsadresse des Image-Kopfes des Boot-Files
!                       verwendet. Andernfalls wird die Steuerung zu dem ersten
!                       Byte des Boot-Files uebertragen.
!
!                7      Speichertestverbot. Dieses Flag verbietet das Testen
!                       des Speichers waehrend des Bootstrap-Prozesses.
!
!                8      Filename. Veranlasst den Bootstrap-Prozess den Namen
!                       des sekundaeren Boot-Files anzufordern.
!
!                9      Halt vor der Uebertragung. Veranlasst einen Halte-
!                       befehl in Prioritaet gegenueber der Uebertragung
!                       zum Boot-File auszufuehren. Diese Option ist bei
!                       einer Debug Absicht sinnvoll.
!
!               10      Keine PFN Loeschung.
!
!               11      Spezifiziert, dass der Mehrkanalspeicher fuer die
!                       gesamte Ausfuehrung des Speicherbedarfes zu verwenden
!                       ist.
!                       Kein lokaler Speicher wird verwendet. Wenn Bit <5>
!                       auch gesetzt ist, tritt der Diagnosesupervisor in den
!                       "Autotest"-Modus ein.
!
!               12      Spezifiziert, dass der Mehrkanalspeicher in Ver-
!                       bindung mit dem lokalen Speicher als ein Speicher ver-
!                       wendet werden kann.
!
!               13      Spezifiziert weitere Exhaustive-Tests des Speichers,
!                       damit nichtkorrigierbare Fehler lokalisiert werden
!                       koennen.
!
!           <31:28>     Spezifiziert die Nummer der obersten Ebene der
!                       Directory auf der Systemplatte mit mehrfach operieren-
!                       den Systemen.
!
!        SP     Adresse+(%X200) der ersten arbeitenden 64 Kb Speicherregion,
!               verwendet als Stack-Pointer und Pointer zum Speicher.
```

```
!
!
! Ausgabeparameter:
!
!        R10             - Die Basisadresse dieser Region beinhaltet den
!                          sekundaeren Bootstrap-Prozess.
!        R11             - Pointer zum Restart-Parameter-Block (RPB)
!        SP              - Stack-Pointer
!        PR$_SCBB        - Systemsteuerblockbasisregister
!
!        Skizze des Speichers zum Start des sekundaeren Bootstrap-Prozesses:
!
!        +-------------------------------------+ :Basis
!        !                                     !
!        !      Restart-Parameter-Block (RPB)  !
!        !                                     !
!        +-------------------------------------+ :Basis+%X200
!        !                                     !
!        !         primaerer Bootstrap-Code    !
!        !                                     !
!        !                                     !
!        +-------------------------------------+ :PR$_SCBB
!        !                                     !
!        !            Systemsteuerblock        !
!        !                                     !
!        +-------------------------------------+ :PFNMAP
!        !                                     !
!        !               PFN Bitplan           !
!        !                                     !
!        +-------------------------------------+ :PFNMAP+%X800
!        !                                     !
!        !             Bootstrap-Stack         !
!        !                                     !
!        +-------------------------------------+ :(SP)
!        !                                     !
!        !        sekundaerer Bootstrap-Code   !
!        !                                     !
!        +-------------------------------------+
!
!<Ende-von-BOOT.HLP>
```

3.11. Helpfile ERROR.HLP

```
!Revision 1.1
!Fuer Systeme mit 5 1/4"-Floppylaufwerken
!
!          Helpfile fuer Fehlermeldungen der Konsole
!          ------------------------------------------
!
! Diese Datei listet alle Fehlernachrichten auf, gibt Ursachen
! und moegliche Korrekturverfahren an.
!
!=================================
! syntaktische Fehler :          !
!=================================================================
! ?'<text-zeichenkette>' IS      !  Die <text-zeichenkette> ist kein
! INCOMPLETE                     !  vollstaendiges Konsolkommando.
!-----------------------------------------------------------------
! ?'<text-zeichenkette>' IS      !  Die <text-zeichenkette> wurde nicht
! INCORRECT                      !  als gueltiges Kommando anerkannt.
!-----------------------------------------------------------------
! ?FILE NAME ERR                 !  Ein <filename>, welcher mit einem Kom-
!                                !  mando gegeben wurde, kann nicht in
!                                !  RAD50 uebersetzt werden.
!                                !  (<filename> ist ungueltig.)
!-----------------------------------------------------------------
! ?IND-COM ERR                   !  In dem Format eines indirekten Kom-
!                                !  mandofiles wurde von der Konsole ein
!                                !  Fehler gefunden.
!                                !  Moegliche Fehler sind:
!                                !  1) Mehr als 80 Zeichen in einer in-
!                                !     direkten Kommandozeile, oder
!                                !  2) die Kommandozeile endet nicht mit
!                                !     <CR> und <LF>.
!=================================================================
! Kommando-Fehler:               !
!=================================================================
! ?DXi NOT ENABLED               !  Der logische Name 'DXi' ist keinem
!                                !  Floppylaufwerk zugewiesen (Mit 'SHOW'
!                                !  pruefen, ggf. 'ENABLE DUAL' eingeben).
!-----------------------------------------------------------------
! ?FILE NOT FOUND                !  Ein File mit dem Namen <filename> ist
!                                !  auf der Diskette, die auf dem mit <dev>
!                                !  angesprochenen Laufwerk liegt, nicht
!                                !  vorhanden.
!-----------------------------------------------------------------
! ?NO CPU RESPONSE               !  Konsol-Timeout beim Warten auf eine
!                                !  Antwort von der CPU. (Ein Wiederholen
!                                !  zeigt moegliche CPU-bezogene Hard-
!                                !  warefehler.)
!-----------------------------------------------------------------
! ?CPU NOT IN CONSOLE WAIT LOOP, !  Ein Konsolkommando fordert eine
!  COMMAND ABORTED               !  Unterstuetzung von der CPU an,
!                                !  wenn die CPU sich nicht in
!                                !  der Konsol-Serviceschleife befindet.
!                                !  (Kommando 'HALT'; Kommandowiederholung)
!-----------------------------------------------------------------
```

?CPU CLK STOP, COMMAND ABORTED	Ein den CPU-Takt zum Laufen auffor- derntes Konsolkommando wurde bei Stepmodus angewiesen. (Stepmodus loeschen; Wiederholung)
?CANT DISABLE BOTH FLOPPIES, FUNCTION ABORTED	Es wurde versucht den lokalen Floppy und den Remote-Floppy zu sperren.
Fehler, erzeugt durch Floppies :	
?FLOPPY ERR, CODE=X	Der Konsolfloppy-Driver erkannte einen Fehler. Code=1 Floppy_Hardware-Fehler (CRC, Paritaet, etc.) Code=2 Der File wurde nicht gefunden. Code=3 Ueberlauf der Floppy-Driver- warteschlange. Code=4 Konsolsoftware forderte eine illegale Sektorennummer an.
?FLOPPY NOT READY	Beide Floppies nicht ansprechbar. (Fehler der AFS, Ausfall der Strom- versorgung)
?FLOPPY NOT READY DRIVE_x	Das Laufwerk x ist nicht installiert oder offen, oder , es liegt keine Diskette drin.
?B-NO CONSOL.SYS	Das Directory der zum Laden einge- legten Diskette enthaelt kein File mit dem Namen CONSOL.SYS. Es er- folgen automatisch unbegrenzt viele Leseversuche.
?B-I/O-ERROR	Hardwarefehler beim Laden des CONSOL.SYS (automatische Wiederholung)
?NO BOOT ON FLOPPY DRIVE_x	Die Diskette auf dem Laufwerk x enthaelt keinen gueltigen Boot-Block (bei x=0 automatisch Uebergang zu Laufwerk 1, bei x=1 wird zusaetzlich ein Diskettenwechsel angefordert).
?FLOPPY ERROR ON BOOT DRIVE_x	Floppyfehler beim Lesen des Absolut- laders durch den Anfangslader (Wieder- holung nach Eingabe irgendeines Zeichens ueber Tastatur).
Mikroroutinen Fehler :	
?MIC-ERR ON FUNCTION	Ein Mikrofehler trat in der CPU waehrend der Bearbeitung einer Konsolanforderung auf. Nachdem diese Nachricht gegeben wurde, wurden die ISB-Fehlerregister zwischengespeichert. (Handlungsweise ist vom Fehler

	abhaenging.)
?INT-REG ERR	Ein Mikrofehler trat waehrend des Versuches einer Referenz zu einem internen Register auf. Die Ursache dieses Fehlers wird eine illegale Adresse sein.
?INT-REG ERR	Ein Mikrofehler trat waehrend des Versuches eines Zugriffes auf ein internes Register auf. Die Ursache dieses Fehlers kann eine illegale Adresse sein.
?MICRO-ERR, CODE=X	Ein nicht anerkannter Mikrofehler trat auf. Der Code, der von der CPU zurueckgefuehrt wurde, ist kein bekannter Fehlercode. 'X' ist der von der CPU zurueckgefuehrte Code.
?MEM-MAN FAULT,CODE=XX	Ein 'EXAMINE' oder 'DEPOSIT' mit virtueller Adresse verursachte einen Fehler in der Speicherverwaltungsroutine. 'XX' ist ein Ein-Byte-Fehlercode welcher durch die Routine zurueckgefuehrt wurde. Bitzuordnungen: Bit 0 = Laengenverletzung (Bits von rechts nummeriert) Bit 1 = Fehler war in einer PTE-Referenz. Bit 2 = Write- o. Modify-Versuch Bit 3 = Zugriffsverletzung Bit 4 bis 7 koennen ignoriert werden.
CPU-Fehler	
?INT-STK INVLD	CPU-Unterbrechung auf Grund ungueltig makierten Interrupt-Stacks.
?CPU DBLE-ERR HLT	Ein Maschinenfehler trat auf, bevor ein vorausgegangener behandelt wurde. Dies ist die Ursache, dass die CPU ein 'DOUBLE ERROR'-Halt abarbeitet. (Pruefen der ID-Register 30-3F (HEX). Daten zeigen Ursache der Pruefung).
?ILL I/E VEC	Die CPU erkannte einen illegalen Interrupt/Exception-Vektor.
?NO USR WCS	Die CPU erkannte einen Interrupt/Exception-Vektor zum Nutzersteuerspeicher (UCS), aber es existiert keiner.
?CHM ERR	Ein Modusaenderungsbefehl wurde von

	dem Interrupt-Stack versucht.
INT PENDING	Eigentlich ist dies kein Fehler, zeigt aber an, dass ein Fehler in der Zeit, als ein von der Konsole gefordertes 'Halt' ausgefuehrt wurde, moeglich war. Loeschen des Interrupts.
?MICRO-MACHINE TIME OUT	Zeigt an, dass der RVS-K 1840 nicht in der Lage war, Interruptsignale in der zulaessigen Zeit zu uebernehmen.
Kompatibilitaets-Fehler :	
?WARNING-WCS & FPLA VER MISMATCH	Der Mikrokode im WCS ist nicht mit FPLA kompatibel. Diese Mitteilung wird bei jedem CPU-Start oder jedem Continue ausgegeben. Es werden aber keine anderen Handlungen durch die Konsole entgegengenommen.
?FATAL-WCS & PCS VER MISMATCH	Der Mikrokode im PCS ist nicht kompatibel mit dem im WCS. Ein CPU-Start und eine Fortsetzung sind durch die Konsole verhindert.
?REMOTE ACCESS NOT SUPPORTED	Wird ausgegeben, wenn sich der Konsolmodusschalter in der 'REMOTE'-Position befindet, und die Remote-Unterstuetzungssoftwareroutinen nicht in die Konsole eingebunden wurden.
Konsol-Fehler :	
?TRAP-4, RESTARTING CONSOLE	Die Konsole ging in den Timeout-Trap. Die Konsole veranlasst einen Restart.
?UNEXPECTED TRAP MOUNT CONSOLE FLOPPY, THEN TYPE ^C	Die Konsole sprang zu einem unbenutzten Vektor. Konsole wird erneut geladen, wenn Control/C eingegeben wird.
?Q-BLKD'	Die Ausgabewarteschlange des Konsolterminals ist blockiert. Die Konsole wird erneut geladen.

<Ende-von-ERROR.HLP>

4. Mikrodiagnose

4.1. Zusammenfassung

Die Mikrodiagnose ist ein Programmpaket der Ebene 5 (standalone Modus). Sie ist bestimmt zur Überprüfung der prinzipiellen Funktionsfähigkeit der einzelnen Funktionsgruppen (BLP's) der zentralen Verarbeitungseinheit (CPU), des Hauptspeichers (MEM), und bedingt des SKRBUS-Adapters (SBA) sowie des MSBUS-Adapters (MBA). Sie ist darüber hinaus zur Fehlersuche und Fehlerbeseitigung auf diesen Funktionsgruppen in der Entwicklung, Produktion, im Vertrieb und im Kundendienst einzusetzen. Vorausgesetzt wird die volle Funktionsfähigkeit des Konsolsubsystems, d.h. die fehlerfreie Abarbeitung der entsprechenden Konsoltestprogramme.

Zur Mikrodiagnose gehören folgende Dateien :

- PSKAB.SYS (MICMON.SYS)	(Dok.Nr.: 1.57.700301.2)	Diagnose-Monitor (DM)
- PSKAC.TSK	(Dok.Nr.: 1.57.700302.0)	Hardcoretest-Monitor (HM)
- PSKAD.TSK	(Dok.Nr.: 1.57.700303.7)	Hardcoretests (HT)
- PSKAE.TSK	(Dok.Nr.: 1.57.700304.5)	Mikrotest-Monitor (MM)
- PSKAF.TSK	(Dok.Nr.: 1.57.700305.3)	Kommandoeingabe-Programm (KP)
- PSKAG.TSK	(Dok.Nr.: 1.57.700306.1)	Directory-Programm (DI)
- PSKAK.TSK	(Dok.Nr.: 1.57.700307.8)	Mikrotestfehler-Monitor (FM)
- PSKAH.TSK	(Dok.Nr.: 1.57.700308.6)	Mikrotest 1 (MT1)
- PSKAL.TSK	(Dok.Nr.: 1.57.700309.4)	Mikrotestfehler 1 (MF1)
- PSKAR.TSK	(Dok.Nr.: 1.57.700310.0)	Mikrotest 2 (MT2)
- PSKAS.TSK	(Dok.Nr.: 1.57.700311.7)	Mikrotestfehler 2 (MF2)

4.2. Testbedingungen

4.2.1. Hardware

- vollfunktionsfähiges Konsolsubsystem K 2811
- CPU K 1840 (KAA40)

4.2.2. Software

- Diskette mit Standard-Konsolprogramm	DS1
- Diskette mit Mikrodiagnose 1	DS2
- Diskette mit Mikrodiagnose 2	DS3

4.3. Voraussetzungen

Siehe Pkt. 4.2.1 und 4.2.2

4.4. Bedienungshinweise

Die Mikrodiagnose ist mit Hilfe des Konsolprogrammes nach Einlegen der Diskette DS2 oder DS3 in die FSE des Konsolrechners durch Eingabe von " T " sofort zu starten (s. Beschreibung Konsolsteuerung, Kap.3) bzw. durch Eingabe von " T/C " zu laden, um zunächst in den Kommandoeingabemodus der Mikrodiagnose überzugehen. Siehe hierzu Pkt. 4.6.3.2.

4.5. Funktionsbeschreibung

4.5.1. Übersicht

Die Mikrodiagnoseprogramme dienen der Überprüfung der Funktionsfähigkeit des Hauptrechners (HR) im System K 1840. Sie ermöglichen die Bestimmung fehlerhafter Funktionsgruppen innerhalb der CPU, Gleitkommaarithmetik und Speichersteuerung. Bei erkannten Fehlern erfolgt am Konsolterminal eine Ausschrift mit Angabe der Steckeinheit bzw. Steckeinheitengruppe, der der Fehler zuzuordnen ist.

Das Mikrodiagnosesystem besteht aus zwei hauptsächlichen Teilen:
- Konsoladapter- und Hardcore-Tests (HT)
- Mikrotests (MT).

Jeder Teil wird durch einen entsprechenden Monitor gesteuert:
- Hardcoremonitor (HM)
- Mikrotestmonitor (MM).

Beide Testteilmonitore werden wiederum durch einen übergeordneten und im Speicher des Konsolrechners während der Mikrodiagnose residenten Mikrodiagnostikmonitor (DM) gesteuert. Die Monitore und Testprogramme befinden sich auf Disketten und sind von der Folienspeichereinheit (FSE) des Konsolsubsystems (KSS) über das Konsolprogramm durch Eingabe von "T" zu laden. Durch Setzen des Schalters "/C" wird nach dem Laden der Mikrodiagnose in den Kommandomodus gegangen, ansonsten wird sofort mit der Abarbeitung der Testfolge begonnen.

Die prinzipielle Testfolge ist - Hardcoretests
- Mikrotests.

4.5.2. Programmgröße (Speicherplatzbedarf und Speicheraufteilung)

Die Hardcoretests überprüfen die Funktionsfähigkeit einer Minimallogik, die zur zuverlässigen Ausführung der Mikrotests erforderlich ist, d.h. Grundhardware zum Datentransfer und zur Fehlerermittlung und -anzeige. Zur Gewährleistung der Abarbeitung der Mikrodiagnose auch bei einer minimalen Speicherkonfiguration des KSS wird beim Laden der Mikrodiagose das Konsolprogramm überschrieben. Der Stack, auf den alle Programmteile zugreifen, befindet sich im Bereich von '776 (alle Adressen oktal) abwärts. Der DM wird in den Bereich von '1000 bis '6366 geladen und ist hier resident. Im Bereich von '6370 bis '6566 befinden sich globale Variable, die den aktuellen Stand des Diagnoseablaufes beinhalten und auf

die alle Programmteile zugreifen. Bei Abarbeitung der Hardcoretests wird der HM in den Bereich von '6570 bis '16000 und die jeweils aktuellen HT in den Bereich von '20400 bis '25400 (minimal 128 Byte, max. 12 X 128 Byte) geladen, und zwar in Form von Pseudobefehlen, die die entsprechenden Anweisungen (s.Pkt. 2.2) wiederspiegeln und im wesentlichen auf Subroutinen des DM zurückgreifen.

Bei Abarbeitung der MT wird der MM in den Bereich von '6570 bis '20376 geladen. Die MT befinden sich auf der Diskette in einer komprimierten Datenstruktur und werden in dieser Form in den Bereich von '22000 bis '22600 (3 X 128 Byte) geladen, vom MM dekomprimiert in den Bereich von '20400 bis '21776 und dann in den WCS des HR geladen, um von dort aus abgearbeitet zu werden. Im Fehlerfall bei der Abarbeitung von MT wird durch den DM ein Fehlermonitor (FM) zur Fehlerisolierung in den Bereich von '6570 bis '12510 geladen und damit der MM teilweise überschrieben.

Ist beim Start der Diagnose über das Konsolprogramm der Schalter "/C" gesetzt worden oder wurde bei der Testabarbeitung ein Fehlerhalt angewiesen bzw. der Testablauf durch einen Bedienerinterrupt (CTRL/C) unterbrochen, so wird durch den DM von der Diskette das Kommandoprogramm (KP) in den Bereich von '7370 bis '20170 geladen, wodurch der jeweils aktuelle Testteilmonitor überschrieben wird. Wird bei der Kommandoeingabe eine bestimmte Test- oder Sektions-Nr. angegeben, so bedient sich das Kommandoprogramm eines Directory-Programms (DI), um zu ermitteln, ob sich der betr. Test, bzw. die betr. Sektion auf der eingelegten Diskette befindet. Der Fehlermonitor benutzt zur Fehlerisolierung Fehlerprogramme (MF), die jeweils den entsprechenden Tests zugeordnet sind und bei Bedarf von der Diskette in den Pufferbereich geladen werden.

Die für die im KSS abzuarbeitenden Programme erforderlichen Driver (Terminal und FSE) befinden sich resident im ROM-Bereich von '140000 bis '157776 und werden über Emulator-Traps angesprungen.

```
          Floppy- und Terminal- Driver
                  |
                  |
 ------------------------------------------------------
 |                        |                           |
 Konsolprogramm          DM                WCS-Debugger
                          |
 ----------------------------------------------
 |       |       |                |             |
 DI<---->KP      HM               MM           FM
                  |                |             |
                  |                |             |
                  HT               MT-Puffer   MF-Puffer
```

Abb.1: Programmresidenz im Speicher des KSS
Die auf einer Zeile dargestellten Programmteile belegen etwa die gleichen KSS-Hauptspeicherbereiche, werden durch den DM jeweils entsprechend des Programmablaufes überlagert, schließen sich also gegenseitig im Speicher aus.

4.5.3. Programmlaufzeit

Die fehlerfreie Abarbeitungszeit , einschließlich des Ladens der entsprechenden Monitore und Tests von der Konsoldiskette beträgt:

- Hardcoretests 120 s
- Mikrotests 1 120 s
- Mikrotests 2 190 s

4.5.4. Abarbeitungsbedingungen

entfällt.

4.5.5. Flag-Verwendung

entfällt.

4.5.6. Allgemeine Programmanforderungen

entfällt.

4.5.7. Fehlerermittlung

Die Fehlerermittlung erfolgt auf der Basis der vom DM erzeugten Fehlermitteilungen unter Zuhilfenahme der Assemblerlisten der Testprogramme (HT, MT1, MT2).

Fehlermitteilungen sind Ausschriften auf dem Konsolterminal, die mit einem "?" beginnen. Siehe hierzu Pkt. 4.6.3.3.

4.5.8. Programmitteilungen

Wie Fehlermitteilungen, jedoch ohne "?". Siehe hierzu Pkt. 4.6.4.1.

4.5.9. Leistungsparameter des Testobjektes

entfällt.

4.5.10. Aktionen bei unerwarteten Fehlern

entfällt.

4.5.11. Testfolge bei mehreren Einheiten

Es kann auch bei Rechnerverbundsystemen jeweils immer nur eine CPU getestet werden.

4.5.12. Einstellung für Ablauf unter AFT

entfällt.

4.5.13. Testsektionen

Die gesammte Diagnose wird unterteilt in fortlaufend hexadezimal von 001 bis 2A6 durchnummerierte Tests von denen jeweils eine bestimmte Anzahl zu Sektionen zusammengefaßt sind, die ebenfalls hex. von 01 bis 66 durchnummeriert sind Die Tests von 001 bis 03F, bzw. Sektionen von 01 bis 1F sind Hardcoretests, alle übrigen sind Mikrotests. Wird nicht durch entsprechende Kommandos etwas anderes angewiesen, so werden die Tests in der Reihenfolge ihrer Durchnummerierung abgearbeitet. Nach dem Laden eines Tests von der Diskette wird vor der Testausführung am Konsolterminal die Sektions-Nr. ausgegeben.

4.6. Testbeschreibung

4.6.1. Hardcoretest

4.6.1.1. Allgemeine Beschreibung

Die Hardcoretests bilden den Anfang der auszuführenden Mikrodiagnostiktests. Ihre fehlerfreie Abarbeitung ist Voraussetzung für die Abarbeitung der nachfolgenden Mikrotests. Die Hardcoretests prüfen zunächst die Steuer- und Datenregister des Consol Interface Board (CIB). Dies sichert den Zugriff zum internen Datenbus des Hauptrechners (IDBUS).
Nach dem CIB-Test erfolgt eine Prüfung der Taktzentrale des HR (Clock Board). Über den VBUS wird das Ein- und Ausschalten des Taktgenerators, Einzeltaktbetrieb sowie der Status verschiedener Taktfunktionen abgetestet. Weiterhin wird die fehlerfreie Abarbeitung von Mikrobefehlsfolgen geprüft. Es werden z.B. Daten in den Mikrostack und zurück übertragen, Adressen in den Mikrostack geladen und durch RETURN diese Adressen aus dem Mikrostack in den Mikrobefehlszähler (UPC) geladen. Diese Strategie gestattet es, die Mikroadreßkanäle in kleinen Schritten auszutesten.
Die übrigen Hardcoretests überprüfen dann im wesentlichen den schreibbaren Mikrobefehlsspeicher (WCS), den PROM-Mikrobefehlsspeicher (PCS) sowie die Grundelemente der verschiedenen Datenwege. Hierbei handelt es sich im Prinzip um vollständige Adreß- und Paritätsprüfungen. Die Datenkanäle werden durch Schreiben, Lesen und Soll/Ist-Vergleich mit verschiedenen Registern abgeprüft. Ferner wird die Grundeigenschaft des Rechenwerkes (ALU), Daten zu übertragen und zu vergleichen sowie der Datenerhalt in den Zwischenregistern, die im Fehlerfall zum Abspeichern der Fehlerdaten dienen, abgetestet.

```
HT----CIB, (BLP 8929) ----CLOCK,(BLP 8916) (Start/Stop, Status)
      Steuer-u.|
      Daten-   |
      Reg.,    |---USC, (BLP 8923) (Adreßltg.,Traps,
      IDBUS,   |        ECO's,Return)
      Takt     |
               |---WCS, (BLP 8920) (vollst. Adr.- u.Dat.-
               |        Prüf., PBT-Erzeug.)
               |---PCS, (BLP 8922) (vollst.Adr.- u. PBT-
               |        Prüfung)
               |---Datenkanäle (Q-, D-Reg.
                        Zwischenreg., ALU)
```

Abb.:2 Hardcoreteststruktur

4.6.1.2. Struktur der Hardcoretests

Der nachfolgende Abschnitt beschreibt die Pseudobefehle der Hardcoretests und die dazugehörenden Anweisungszeilen in den Listings der Hardcore-Tests. Diese Erläuterungen sollen dem Verständnis und der Interpretation der Listings dienen. Dabei werden folgende Symbole und Abkürzungen benutzt:

< > Kennzeichnung von Kategorien, Namen oder Argumenten einer Anweisungszeile.

[ ] Kennzeichnung, daß dieser Teil der Anweisungszeile wahlfrei ist, d.h. angegeben werden kann oder auch nicht.

, Trennen von Kategorien, Namen oder Argumenten innerhalb einer Anweisungszeile.

SRC Abkürzung für Quelle (source).

DST Abkürzung für Senke (destination).

I,J,K legale Indexnamen.

Diese Anweisungen stellen Pseudobefehle dar, die zwecks Speicherplatzeinsparung auf Subroutinen im HM zurückgreifen.

BLKMIC <SRC ADDR>,<SRC INDEX>,<WCS ADDR>,<WORDCOUNT>,
[<WCS ADDR INDEX>]
Blocktransfer
Transportiere die <WORDCOUNT> Anzahl von 96-Bit-Mikroworten von <SRC ADDR>, indexiert mit <SRC INDEX> in den WCS ab <WCS ADDR>, ggf. indexiert mit <WCS ADDR INDEX>. Wenn <SRC INDEX> angegeben ist, bezieht er sich auf 6 KSS-Worte, d.h. 96 Bit. Beginnt <WCS ADDR> mit einem Buchstaben, so stellt er den Zeiger in einer Tabelle des Testdatenbereiches dieses Tests dar, ansonsten ist es eine physische WCS-Adresse.

CHKPNT [<PASS ADDR>],[<FAIL ADDR>]
Testverzweigung
Wenn das während einer Vergleichsanweisung gesetzte Fehlerflag = 0 ist, verzweige zu <PASS ADDR>, ist es ungleich 0, verzweige zu <FAIL ADDR>. Ist keine der beiden Adressen angegeben, so wird zur nächsten Anweisung übergegangen und die Adresse der nächsten Anweisung ausgegeben. Sie erscheint auf der Zeile, die mit TRACE bezeichnet ist.

CLOCK <TIMES>
Führe <TIMES> Einzelsystemtakte aus. Ist <TIMES> ein Vielfaches von 4 , so wird alle 4 x <TIMES> ein Buszyklus ausgeführt.

CMPCA [<MODE>],<REG>,<DST ADDR>,[<DST ADDR INDEX>]
Vergleiche Register mit Speicher.
Vergleiche das durch <REG> angegebene KSS-Register mit dem Inhalt der durch <DST ADDR>, indexiert mit <DST ADDR INDEX> angegebenen KSS-Speicheradresse. Das mit <MODE> angegebene Argument ist im allg. Gleichheit (EQUAL), d.h. Standard. Wenn die Vergleichsbedingung nicht erfüllt ist, so wird das Fehlerflag gesetzt. Ist als <REG> IDREGLO oder IDREGHI angegeben, so wird das IDBUS-Register zum Vergleich genommen, das als letztes durch eine READID-Anweisung gelesen wurde.

CMPCAD [<MODE>],<REG>,<DST ADDR>,[<DST ADDR INDEX>]
Vergl. 2 Register mit 2 Speicherplätzen.
Vergleiche <REG> und <REG>+2 mit dem Inhalt von <DST ADDR> und <DST ADDR>+2, indexiert mit <DST ADDR INDEX>. Sonstiger Ablauf wie bei CMPCA.

CMPCAM [<MODE>],<REG>,<MASK ADDR>,[<MASK ADDR INDEX>],
<DST ADDR>,[<DST ADDR INDEX>]
Vergl. maskiertes Register mit Speicherplatz.
Vergleiche das mit dem Inhalt von <MASK ADDR> , ggf.mit <MASK ADDR INDEX> indexiert, maskierte <REG> mit dem Inhalt von <DST ADDR>, ggf. mit <DST ADDR INDEX> indexierten Speicherplatz Die Maskierung erfolgt durch Komplementierung der Maske und Bitlöschen im Register.

CMPCMD [<MODE>],<REG>,<MASK ADDR>,[<MASK ADDR INDEX>],
<DST ADDR>,[<DST ADDR INDEX>]
Vergl. 2 mask. Register mit 2 Speicherpl.
Die Inhalte von <REG> und <REG>+2 maskiert mit den Inhalten von <MASK ADDR> und <MASK ADDR>+2 , werden mit <DST ADDR>, ggf. indexiert mit <DST ADDR INDEX> verglichen. Bei Nichterfüllung entsprechend <MODE> wird das Fehlerflag gesetzt. Nicht angegebener <MODE> bedeutet Gleichheit (EQUAL). Ist als <REG> IDREGLO oder IDREGHI angegeben , so wird das bei der letzten READID-Operation gelesene Register für den Vergleich herangezogen. Die Verwendung der Maske erfolgt wie bei CMPCAM

CMPPCSV <DST ADDR>,[<DST ADDR INDEX>]
Vergl. PC-Rettungsregister mit Speicherinhalt.
Der Inhalt des PC-Rettungsregisters wird mit dem durch <DST ADDR>, ggf. mit <DST ADDR INDEX> indexiert, angegebenen Speicherplatzinhalt verglichen. Bei Ungleichheit wird das Fehlerflag gesetzt.

ENDLOOP <INDEX NAME>
Der Wert von <INDEX NAME> wird inkrementiert (s. LOOP-Anweisung) und zu dem aktuellen Wert, der durch <INDEX NAME> indexspezifiziert ist, addiert. Es wird der aktuelle Wert mit dem letzten durch die LOOP-Anweisung angegebenen Wert verglichen. Ist der aktuelle Wert kleiner oder gleich dem letzten, so wird zu der auf die LOOP-Anweisung folgenden Anweisung übergegangen.

ERLOOP Die Adresse der nächsten Anweisung wird gerettet. Wurde ein Fehler erkannt und ist das LOOP-on-ERROR-Flag gesetzt, so erfolgt nach Ausführung der IFERROR-Anweisung ein Restart von dieser Adresse.

FETCH <WCS ADDR>,[<WCS ADDR INDEX>],[<WCS ROM NOP>]
Ist <WCS ADDR> ein numerischer Ausdruck, so wird ein RETURN ausgeführt zu <WCS ADDR>, ggf. indexiert mit <WCS ADDR INDEX>. Ist <WCS ADDR> ein alphanumerischer Ausdruck, so erfolgt dieser Rücksprung zum Inhalt dieser Adresse. Ist <WCS ROM NOP> angegeben, so wird während des Rücksprunges das Bit 7 im "machine control register" (MCR) gelöscht.

FLTONE <DST ADDR>,<INDEX NAME>
In ein 32-Bit-Wort = 0 wird in die Bitposition eine "1" eingefügt, die durch den aktuellen Wert -1 von <INDEX NAME> angegeben wird. Dieses Wort wird auf <DST ADDR> und <DST ADDR>+2 geladen.

FLTZRO <DST ADDR>,<INDEX NAME>
Die Ausführung erfolgt sinngemäß wie bei FLTONE nur, daß in ein 32-Bit-Wort mit alles "1" eine "0" eingefügt wird.

IFERROR <MESSAGE NUMBER>,[<FAIL ADDR>]
Fehlerbehandlung
Ist das Fehlerflag ungleich 0, so erfolgt auf dem Konsolterminal die Ausgabe folgender Testparameter: PC der Anweisung, Test-Nr., Subtest-Nr., Soll- und Ist-Daten. Bei nicht gesetztem HALTD-Flag wird zu <FAIL ADDR> gesprungen. Ist das Fehlerflag = 0 oder <FAIL ADDR> nicht angegeben, so wird zur nächsten Anweisung übergegangen.

INITIALIZE
Setzen und Löschen des CPU-Initialisierungsbits.
Einzeltaktbit (single time state) löschen.
Einzelbuszyklusbit (single bus cycle) setzen, ROM NOP-Bit setzen, Fortsetzungsbit (proceed) im MCR setzen.

KMXGEN <SRC ADDR>,<INDEX NAME>
Durch den aktuellen Wert - 1 von <INDEX NAME> wird die KMUX-Adresse erzeugt und auf <SRC ADDR> des KMUX-Feldes der Mikrobefehle geladen. <SRC ADDR> zeigt auf eine 6-Wort-Tabelle des Testdatenbereiches, die den Mikrobefehl enthält.

LDIDREG <REG>,<SRC ADDR>,[<SRC ADDR INDEX>]
Laden IDBUS-Register.
Das durch <REG> angegebene IDBUS-Register wird mit dem Inhalt von <SRC ADDR> und <SRC ADDR>+2 ggf. indexiert mit <SRC ADDR INDEX>, geladen. Ist <REG> eine Mikrostack-, Mikrobreak- oder WCS-Adresse, so werden als Inhalt von <SRC ADDR> 16 Bit geladen, ansonsten 32 Bit.

LOADCA <REG>,<SRC ADDR>,[<SRC ADDR INDEX>]
Laden Konsolregister.
Das mit <REG> angegebene Register des KR wird mit dem Inhalt von <SRC ADDR>, ggf. indexiert mit <SRC ADDR INDEX> geladen (16 Bit).

LOOP <INDEX NAME>,<START>,<END>,[<SIZE DEPENDENT>]
Initialisieren einer Testschleife.
Der durch <INDEX NAME> angegebene LOOP-Parameter wird auf den durch <START> angegebenen Wert initialisiert. Der mit <END> angegebene Wert wird für die ENDLOOP-Anweisung gerettet. Die Berechnung des Inkrements erfolgt nach folgenden Algorithmus:
Ist <START> kleiner oder gleich <END>, so wird der Wert um 1 erhöht, ansonsten um 1 erniedrigt. Ist <END> ein <INDEX NAME>, so ist der aktuelle Wert dieses Indexnamens zu retten. Ist <SIZE DEPENDENT> angegeben und gibt es im System nur einen WCS-Modul, so ist der größere Wert von <START> bzw. <END> durch 2 zu teilen, ansonsten nicht. Die Tests sind für 2 WCS-Module geschrieben. Dieses Argugument gestattet es, im Programm die LOOP-Parameter zu ändern, wenn das System nur einen WCS-Modul hat.

MASK <DST ADDR>,<MASK ADDR>
Maskieren von Speicherinhalten.
Der Inhalt von <MASK ADDR> wird komplementiert und mit dem Ergebnis eine Bitlöschoperation bezüglich des Inhaltes von <DST ADDR> vorgenommen.

MOVE <SRC ADDR>,[<SRC ADDR INDEX>],<DST ADDR>
Datentransfer
Der Inhalt von <SRC ADDR>, ggf. indexiert mit <SRC ADDR INDEX> wird nach <DST ADDR> transportiert.

NEWTST <TEST NAME>,[<TEST DESCRIPT>],[<LOGIC DESCRIPT>],
[<ERROR DESCRIPT>],[<SYNC POINT DESCRIPT>]
Es wird ein Testkopf (test header document) mit den angegebenen Argumenten erstellt. Die Fehlerflags werden gelöscht und der PC der nächsten Anweisung abgespeichert.

READID <REG>
Lesen des durch <REG> angegebenen IDBUS-Registers und Laden von IDREGLO und IDREGHI mit dessen Inhalt.

RESET Ausführung eines KSS (K 1620)-RESET-Befehles.

REPORT <MODULE NAME STRING>
Ausgabe der Modul-Nr. d.h. des durch <MODULE NAME STRING> angegebenen Ausdruckes. Ist das HALTI-Flag gesetzt, so erfolgt Rückkehr zum Mikrodiagnostikmonitor.

SETPSW <DATA>
Laden des Prozessorstatuswortes im KSS mit <DATA>.

SETVEC <VECTOR ADDR>
Setzen der durch <VECTOR ADDR> angegebenen KSS-Adresse für die erwartete Trap-Routine.

SKIP [<DST ADDR>]
Sprung nach <DST ADDR>. Wenn nicht angegeben, dann Sprung zum nächsten Test. Beginnt <DST ADDR> mit "S", dann erfolgt Übergang zum nächsten Subtest.

SUBTEST Incrementieren des Subtestzählers.

TSTVB <SRC TABLE ADDR>,[<SRC TABLE ADDR INDEX>]
Laden und Lesen VBUS.
Die vom VBUS gelesenen Daten werden mit dem Inhalt von <SRC TABLE ADDR>, ggf. indexiert mit <SRC TABLE ADDR INDEX>, verglichen.
<SRC TABLE> hat folgendes Format:

```
1$:     .WORD   ; Anzahl zu prüfender Bits
        VBUSG   ; Kanal-Nr., Bit-Nr., Bitwert
        "

2$:     .WORD
        VBUSG
        "
        "
        "
```

VBUSG ist ein Makroname, der 3 Argumente enthält in einem 16-Bit-Wort:

```
Bit 00...07     Kanal-Nr.
Bit 08...14     Bit-Nr.
Bit 15          Sollwert
```

TYPSIZE
BADDATA (Aresse) enthält den Wert des gelesenen WCS-Daten-Registers zur Bestimmung der WCS-Modul-Konfiguration. Am Konsolterminal wird die Nr. des getesteten WCS-Moduls ausgegeben. Bei den folgenden Bedingungen wird die Testausführung abgebrochen und das NER-Flag (no error report) gesetzt:

- WCS-Zähler gleich 0
- Bits 00...03 ungleich 0
- 5. K vom WCS nicht vorhanden.

Diese Bedingungen bedeuten, daß der WCS entweder nicht richtig konfiguriert ist oder das WCS-Daten-Datenregister nicht richtig gelesen werden kann.

Die diesen Anweisungen zugeordneten Pseudobefehle bestehen aus:

- 1. Wort, niederw. Byte = Code des Pseudobefehles
  höherw. Byte = Anzahl d. Argumente
- 0 bis 5 nachfolgende Worte = Argumente d. Anweisungen

Diese Pseudobefehle werden zur Abarbeitung der Tests sektionsweise durch den HM von der Diskette in den unter Pkt. 4.5.2. genannten Speicherbereich geladen. Die physischen Adressen der einzelnen Pseudobefehle ergeben sich aus der Anfangsladeadresse ('020400) und den in den Listings angegebenen bzw. im Fehlerfall am Terminal ausgegebenen TPC als Index (s. Pkt. 4.6.3.3.4).

4.6.2. Mikrotest

4.6.2.1. Allgemeine Beschreibung

Bei fehlerfreiem Testdurchlauf überschreibt nach Beendigung der Hardcoretests der DM den HM mit dem MM. Der weitere Testablauf, d.h. die Abarbeitung der Mikrotestfolge, wird dann durch den MM gesteuert.

Der MM beginnt, die auf der Diskette in einer komprimierten Datenstruktur abgespeicherten MT in denselben Pufferbereich des KSS-Speichers zu laden, der auch von den Hardcoretests benutzt wurde. Sie werden dort dekomprimiert und, da sie im Systemmikrocode implementiert sind, in den WCS umgeladen und von dort aus abgearbeitet. Der erste Block enthält eine Tabelle mit den Ansprungadressen der

einzelnen Tests. Diese Adresse wird in den Mikrostack geladen und ein RETURN ausgeführt, um damit den Mikrobefehlszähler (UPC) zu laden und den betreffenden Test zu starten. Am Ende eines jeden Tests wird in den MM zurückgesprungen, der die fehlerfreie Ausführung des Testes abprüft. Der MM initialisiert dann den nächsten Test durch einen erneuten RETURN. Diese Folge wird fortgesetzt bis die aktuelle 1K-Mikrowort-Überlagerung abgearbeitet worden ist. Dann wird durch den MM von der Diskette ein weiteres 1K-Mikrowort in der gleichen Weise in den WCS geladen und dort abgearbeitet. Wegen des Umfanges des Mikrotestpaketes sind 2 Disketten erforderlich. Nach Abarbeitung des letzten Tests einer Diskette überprüft der MM, ob es der letzte Test der gesamten Mikrotestfolge ist. Ist dies der Fall, so wird am Konsolterminal das Durchlaufende ausgegeben, ansonsten eine Aufforderung an den Bediener, die 2. Diskette in die FSE des KSS einzulegen und ein Kommando zur Testfortsetzung einzugeben.

4.6.2.2. Mikroteststruktur

Die ersten Mikrotests vervollständigen den von den Hardcoretests vorgenommenen Test der Datenkanäle. Die Mikrotests beginnen dann, den Übertragungspuffer (translation buffer) TB, sowie den Cache (ohne den verwendeten Speicher) abzuprüfen und zwar auf Adreß-und Datenerhalt sowie auf richtige Paritätsprüfung.
Dann erfolgt eine Testung des Befehlspuffers (instruction buffer) IB, wiederum ohne Speicher. Der IB-Test wird in den Cache geladen. Die Mikrotests bewirken, daß Befehlstestmuster vom Cache erzeugt werden, die die IB-Branch-Funktionen abprüfen und die Datenwege steuern.
Die Interrupt- und Bedingungscodelogik wird in ähnlicher Form abgeprüft, d.h. Testdaten in den Cache geladen und nacheinander von dort abgerufen.
Das nächste Testsegment bezieht sich auf die ISB-Steuerlogik, ihre maintenance-Funktionen und das Speichersystem. Nach Beendigung dieser Tests werden solche Funktionen des TB, Cache und ISB-Subsystems abgeprüft, die abhängig sind von der Zuführung von Daten aus dem Speicher.
In geringem Maße werden SKRBUS-Adapter (SBA) und Massbusadapter (MBA) getestet. Hierbei werden ausgewählte Fehler auf dem ISB erzeugt und die richtige Reaktion der Adapter hierauf überprüft. Als letztes erfolgt ein Test der Gleitkommaarithmetik (floating point accelerator) FPA.

```
MT---------- Datenwege
        |
        |--- TB, Cache (ohne Speicher)
        |
        |--- IB, Branch-Funktionen,
        |        Steuerung Datenkanäle
        |
        |--- Interrupt- und Bedingungscodelogik
        |
        |--- ISB-Steuerlogik,
        |        maintenance-Eigenschaften
        |        Speichersystem
        |--- TB, Cache, ISB
        |        Funktionen, den Speicher betr.
        |
        |--- min. SBA- u. MBA-Tests
        |
        |--- FPA
```

Abb. 3 Mikroteststruktur

4.6.3. Steuerung des Testablaufes

4.6.3.1. Allgemeine Beschreibung

Im allgemeinen Fall wird nach Start der Mikrodiagnostik vom Konsolprogramm nach Eingabe von "T" das gesammte Mikrodiagnostikpaket abgearbeitet. Im Fehlerfall kann vom Bediener durch Setzen des Schalters "/C" beim Start oder durch Eingabe von CTRL/C in den Kommandoeingabemodus gegangen werden, dgl. bei Fehlerhalt. Durch Eingabe zweckmäßiger Kommandos kann dann der gesammte Testablauf entsprechend gesteuert werden.

4.6.3.2. Kommandos und Schalter

Nach Ausschrift der Bedienereingabeaufforderung MIC> können folgende Kommandos und beim Kommando DI die angegebenen Schalter eingegeben werden. Alle Kommandos sind durch <CR> abzuschließen.

4.6.3.2.1. Startkommandos

DI	Start des Testablaufes mit Test-Nr. 001, sofern durch nachfolgende Schalter nichts anderes angewiesen wird. (diagnose)
/TE:n	Test Nr. n wird unbegrenzt ausgeführt. (test)
/SE:m	Sektion Nr. m wird unbegrenzt ausgeführt. (section)
/PA:j	Es werden j Durchläufe aller Sektionen einer Diskette ausgeführt, bei j=-1 unbegrenzt. Nicht anzuwenden nach /TE: oder /SE:. (pass)
/CO	Zusammen mit TE:n oder SE:m Fortsetzen des Testablaufes ab angegebenen Test bzw. Sektion. (continue)
/TE:n m	Abarbeiten der Tests von Nr. n bis Nr. m; ist m<n, dann von Test Nr. n bis Ende.
/SE:n m	Sinngemäß, jedoch bzgl. Sektionen.
CO	Fortsetzen des Testablaufes ohne Änderung des Programmsteuerflags. (continue)

4.6.3.2.2. Ablaufsteuerungskommandos

SE FL HD CL FL HD	Setzen bzw. Löschen Flag Halt bei Fehler ohne Ausschrift vor Fehlerisolierungsroutine (VBUS-Test) (set / clear flag halt on error detection)
SE FL HI CL FL HI	Halt bei Fehler nach Fehlerisolierungsroutine mit Fehlerausschrift (set / clear flag halt on error isolation)
SE FL LO CL FL LO	Ständiges Durchlaufen einer kurzen Fehlertestschleife auch bei Aussetzen des Fehlers, zweckmäßig zusammen mit gesetztem NE (set / clear flag loop)

SE FL NE CL FL NE	Unterdrückung der Fehlerausschrift zweckmäßig zur hardwaremäßigen Fehlersuche (set / clear flag no error report)
SE FL BE CL FL BE	Akustisches Fehlersignal, zweckmässig zusammen mit NE und LO zur Erkennung der Fehlerbeseitigung (set / clear flag bell)
SE FL ER CL FL ER	Bei gesetztem Flag wird nach erkanntem Fehler nach Eingabe von CO der aktuelle Test abgebrochen und die Diagnose mit dem nächsten Test fortgesetzt (set / clear flag erase abort)
CL FL LS	Löschen Flag Wiederholung bestimmte Sektion (clear flag loop on section)
CL FL LT	Löschen Flag Wiederholung bestimmter Test (clear flag loop on test)
SE FL AL CL FL AL	Setzen bzw. Löschen aller Flags (set / clear all flags)
SE SO CL SO	Setzen bzw. Löschen Flag HALT bei Erreichen eines vorgegebenen Test-PC-Standes (set / clear stop on micro-match)
SE SO:n CL SO:n	Halt bei <PC>:=n (set / clear stop on micro-match)
SE FP:m CL FP:m	Eintragen der Adresse m in das FPA-microsync-register (set / clear floating point accellerator)
SE ST ST	Einzeltaktmodus, nächster Takt durch Eingabe von SPACE, Beendigung durch beliebige andere Taste, nur bei MT (set step state)
SE ST BU	Einzelbuszyklus, sinngemäß wie ST (set step bus cycle)
SE ST IN	Einzelbefehlszyklus, sinngemäß wie ST, jedoch nur bei HT (set step instruction)
SE CL FA	größte CPU-Taktrate 10,525 MHz (set clock fast)
SE CL NO	normale CPU-Taktrate 10 MHz (set clock normal)
SE CL SL	geringste CPU-Taktrate 8,925 MHz (set clock slow)
SE CL EX	externer Takt (set clock external)
SH	Ausgabe aller Flags auf Konsolterminal (show)
LO	Löschen HD, HI, Setzen NE, LO und CO (loop)
RE	Rückkehr zum Konsolprogramm (return) Diskette DS1 (Konsolsteuerung) muß in FSE eingelegt sein.

4.6.3.2.3. Prüfkommandos

Wenn es das erste Prüfkommando nach Eintritt in den Monitorkommandomodus ist, so wird der aktuelle Mikrobefehl vor der Ausgabe noch ausgeführt. Bei allen folgenden Ausgaben wird kein Mikrobefehl mehr ausgeführt. Die Inhalte der IDBUS-Register T1...T8 werden bei der Ausführung dieser Prüfkommandos zerstört, außer bei VBUS-Prüfungen. Bei allen weiteren "EX" wird der Takt auf CTP0 geschaltet, bevor das Kommando ausgeführt wird.

EX ID:n	Ausgabe	Inhalt	IDBUS-Register "n"
EX VB:n	-"-	-"-	VBUS-Kanal "n"
EX RA:n	-"-	-"-	RA-Zwischenregister "n"
EX RC:n	-"-	-"-	RC- -"- "n"
EX LA	-"-	-"-	LA-Latch
EX LC	-"-	-"-	LC-Latch
EX DR	-"-	-"-	D-Register
EX QR	-"-	-"-	Q- -"-
EX SC	-"-	-"-	SC- -"-
EX FE	-"-	-"-	FE- -"-
EX VA	-"-	-"-	VA- -"-
EX PC	-"-	-"-	PC
EX IS:<phys.Adr.>	-"-	-"-	physische Adresse
(examine .....)			

4.6.3.2.4. Ladekommandos

Ladekommandos für Register sinngemäß wie die Prüfkommandos, nur zusätzliche Eingabe der zu ladenden Daten "d" (hexadezimal)

DE ID:n d
DE RA:n d
DE RC:n d
DE LA:d
DE LC:d
DE DR:d
DE QR:d
DE SC:d
DE FE:d
DE VA:d
DE PC:d
DE IS:n d
(deposit .....)

4.6.3.3. Fehlerausschriften

Alle Fehlerausschriften beginnen zwecks Unterscheidung gegenüber Informationsausschriften mit "?".

4.6.3.3.1. Syntaxfehler

?USE DIAG COMMAND	Es wurde ein CO-Kommando eingegeben, bevor ein DI-Kommando gegeben wurde.
?INVALID COMMAND	Es wurde ein ungültiges Komm. eingegeben.
?INVALID KEYWORD	Falsches Argument bei einem Kommando.
?NUMBER MUST BE HEX	Eingegebener Zahlenwert war nicht hexadez.

4.6.3.3.2. Systemfehlerausschriften

?OPEN FILE:n	Fehler beim Lesen eines Files v.d. Diskette. n=1 Hardwarefehler FSE n=2 File nicht vorhanden n=3 FSE nicht bereit
?READ SECTOR:n	Fehler beim Lesen von der Diskette. n=1 FSE Hardwarefehler n=3 FSE-Warteschlange voll n=4 ungültige Sektor-Nr.
?KEYBORD ERROR:n	Terminal-Eingabefehler n=5 Terminal-Driver nicht verfügbar (busy) n=7 Terminal-Hardwarefehler
?UNEXPECTED TRAP TO 4 PC=.....	Beim angegebenen PC-Stand erfolgte ein KSS (K 1620)-Trap über Vektor 4 (z.B. ungerade Adresse).

4.6.3.3.3. Abarbeitungsfehler des Monitors

?TIMEOUT IN TEST .... UPC=.....
Der Monitor erhält innerhalb von 4 sec keinen Mikrocoderuf.

?EXECUTION OUT OF SEQUENCE UPC=.... SHOULD BE=.....
Die Tests wurden nicht in richtiger Reihenfolge ausgeführt.

?CLOCK STOPPED UNEXPECTEDLY
Der Takt hält an, ohne daß das SOMM-Bit gesetzt worden ist.

?ILLEGAL MONITOR CALL:.....
Mikroprogramm erzeugt Monitorruf mit falschem Argument.

4.6.3.3.4. Testfehlerausschriften

Die Ausschrift bei aufgetretenem Fehler während eines Tests hat folgendes Format:

```
?ERROR: xxxx TEST: yyyy SUBTEST: zzzz
DATA: aaaaaaaa
      bbbbbbbb
        "
        "
        "
      mmmmmmmm
      nnnnnnnn
      TRACE: oo,pp,...
      FAILING MODULES: 8qqq,8rrr,...
      VBUS-CH/BIT/VAL: nnnn,vvvv,wwww,...
```

Hierbei bedeuten:

- xxxx den PC der IFERROR-Anweisung, bei der der Fehler erkannt wurde. Bei HT ist xxxx 6-stellig oktal (TPC), bei MT 4-stellig hexadez (UPC).
- yyyy die Nr. des Testes, bei dem der Fehler auftrat
- zzzz die entsprechende Subtest-Nr.
- aaaaaaaa bis nnnnnnnn Testdaten, deren Bedeutung in dem Testkopf innerhalb der Listings beschrieben wird.
- oo,pp, usw. ergibt sich aus den Fehlerisolierungsroutinen als Satz der Module, denen der Fehler zuzuordnen ist.
- qqq,rrr, usw. die vom Test ermittelten möglichen Steckeinheiten-Nr. (s. Zeichn.-Nr. der STE) Sie werden in der Reihenfolge der Fehlerwahrscheinlichkeit angegeben. Ist keine eindeutige Zuordnung möglich, wird ggf. der entsprechende Bus (z.B. IDBUS oder CSBUS) ausgegeben.

4.6.4. Interpretation der VBUS-Fehlerausschriften

4.6.4.1. Hardcore-Tests

In den Harcoretests 04, 09, 0C, 1B, 1C, 1D,1E, 1F, 21, 24, 27, 28/1, 29/1, und 3D erfolgt die Ausgabe von erwarteten und empfangenen VBUS-Signalen in den ersten beiden Ausschriften unter "Data" und zwar: Kanal/Bit/Soll-bzw. Istwert.

Beispiel für Fehlerausschriften:

```
MIC>DI/SE:7
HCTEST - V1.0
07,
?ERROR: 000252  TEST: 001F  SUBTEST: 0002
DATA: 1010
      1011
      0E88

TRACE:
FAILING MODULES:  8920

Interpretationen der als falsch erkannten VBUS-Signale:
VBUS-Kanal 1 Bit %X01 Sollwert: 0; Istwert: 1
------> Signal: PCSC PAR ERR 63:32 auf CEHE ist falsch.
```

4.6.4.2. Mikrotests

In den Mikrotests, in denen durch den Mikrotestfehlermonitor VBUS-Signale ausgewertet werden und ein VBUS-Signal als falsch erkannt wird, erfolgt eine gesonderte Ausschrift und zwar: Kanal/Bit/Istwert.

Beispiel für Fehlerauschriften:

```
MTEST1 - V1.0
23,24,25,26,27
?ERROR: 1190  TEST: 00A1  SUBTEST: 0001
DATA: 3FFFFFFC
         "
         "
         "

TRACE: 00, 01, 02
FAILING MODULES: 8903, 8915
VBUS-CH/Bit/VAL: 3130, 5011, 5021, 5071, 5001
```

Interpretation der als falsch erkannten VBUS-Kanalsignale:

```
VBUS-Kanal 3 Bit %X13 ---> Sign. TBMC ENABLE 1A auf TBMC ist "0"
   "   "   5  "  %X01 --->   "   SBHP ID ADDR 2  "  SBLF  "  "1"
   "   "   5  "  %X02 --->   "   SBHP ID ADDR 1  "  SBLF  "  "1"
   "   "   5  "  %X07 --->   "   SBHP ID ADDR 0  "  SBLF  "  "1"
usw.
```

Die den VBUS-Kanalbits zugeordneten Signalnamen sind den entsprechenden Tabellen im Servicehandbuch Rechner zu entnehmen.

4.6.5. Beispiele für Testabarbeitung und Fehlermitteilung

In den Beispielen für Terminalausschriften sind Operatoreingaben unterstrichen und Erläuterungen durch (.....) gekennzeichnet.

4.6.5.1. Fehlerfreier Testdurchlauf

Nachfolgend wird die mögliche Ausschrift bei fehlerfreiem Testdurchlauf (Routinetest, Wartung) dargestellt (Nach Einlesen des Konsolprogrammes und Einlegen der Diskette MICTEST1 in die FSE):

```
>>> T
    -
DIAGMONITOR V1.0
HCTEST - V1.0
01,02,03,04,
NO. OF 1K BANKS OF WCS = 0002
05,06,07,08,09,0A,0B,0C,0D,0F,10,11,12,13,14,15,16,17,
18,19,1A,1B,1C,1D,1E,1F,
MTEST1-V1.0
20,
SYSTEM ID REGISTER = 01384BDA
KEA40 FPLA NOT PRESENT
21,22,23,24,25,26,27,28,29,2A,2B,2C,2D,2E,
2F,30,31,32,33,34,35,36,37,38,39,3A,3B,3C,
END PASS 0001
MOUNT FLOPPY MICTEST 2 & TYPE "DI"
MIC> DI
     --
MTEST2-V1.0
3D,3E,
CPU TR = 00000010
MSA40E 64K CHIP AT TR 00000001
LOWER CONTROLLER MAX ADDRESS+1=00100000
UPPER CONTROLLER MAX ADDRESS+1=00100000
DWA40 AT TR 00000003
RHA40 AT TR 00000008
3F,40,41,42,43,44,45,46,47,48,49,4A,4B,4C,4D,4E,4F,
50,51,52,
MSA40-E/H IO BASE ADDRESS = 20002000
LOWER CONTROLLER MAX ADDR + 1 = 00100000
  BOARD NUMBER = 00000000
        NUMBER OF CRD ERRORS = 00000000
MSA40-E/H IO BASE ADDRESS = 20002000
UPPER CONTROLLER MAX ADDR + 1 = 00100000
  BOARD NUMBER = 00000000
        NUMBER OF CRD ERRORS = 00000000
53,
MSA40-E 64K CHIP AT TR 00000001
  8811  ROMS OK
54,55,56,57,58,59,
STARTING FPA TESTS
5A,5B,5C,5D,5E,5F,60,61,62,63,64,65,66
END PASS 0001
MIC>
(Operatoreingaben sind unterstrichen)
```

4.6.5.2. Hardcoretestschleife

```
Beisp. für Fehlerausschrift:
(Fehlerprovokation durch Ziehen des WCS,   Steckpl. 20)
 MIC> DI/TE:1C
      --------
HCTEST-V1.0
05                                (Sektions-Nr.)
?ERROR: 001324   TEST: 01C  SUBTEST: 1
DATA:   1020
        1021            (Paritätsfehler in der Gr. 00...31)
        0001
        0001            (Bit 00)
TRACE: 001434
FAILING MODULES: 8918, 8920, BUSCS
MIC> SE ST IN   <CR>            (Einzelbefehlsabarbeitung)
     ----------------
MIC> LO         <CR>            (Fehlerschleife)
     --------------
TPC = 001272    <SP>            (LDIDREG USCADR)
                ----
TPC = 001300    <SP>            (LDIDREG USCDAT)
                ----
TPC = 001306    <SP>            (FETCH T1DL1,I) <-------
                ----                            |
TPC = 001316    <SP>            (SYNC4C: TSTVB)         |
                ----                            |
TPC = 001324    <SP>            (IFERROR)               |
                ----                            |
TPC = 001272    <SP>            (usw.)                  |
                ----                            |
TPC = 001300    <CR> >--------------------------------
                ----
MIC> CL FL HI   <CR>            (kein Fehlerhalt)
     ---------------
MIC> SE FL BE   <CR>            (akustisches Signal)
     ---------------
MIC> SE FL NE   <CR>            (keine Fehlerausschrift)
     ---------------
MIC> CO                         (Fortsetzen)
     ---
<CTRL/C>                        (Schleife unterbrechen)
--------
MIC>
```

(Operatoreingaben unterstrichen, Erläuterungen in (...))

Im Test 01C innerhalb der Testsektion 05 erfolgt entsprechend der Beschreibung in den Listings ein Test des CSBUS mit durchlaufender "0" (Subtest 1) durch ein Mikrowort, bzw. durchlaufender "1". Bei der Testausführung erfolgt eine Paritätsprüfung über den VBUS bei SYNC4C. Bei diesem Test werden alle vorhandenen WCS-Module abgeprüft. In dem Beispiel ist die Fehler-PC-Adresse 6-stellig, stellt also eine Oktalzahl dar. Damit handelt es sich um einen HT und zwar im Test 1C, Subtest 1. Dieser Test beginnt in den Listings bei Zeile 2842. Die dem Fehler-PC entsprechende IFERROR-Anweisung befindet sich in Zeile 2866. Derartige Anweisungen folgen gewöhnlich nach einem Test oder einem Vergleich, in diesem Falle nach TSTVB. Es werden also VBUS-Signale verglichen, die in diesem Falle nicht mit den Erwartungswerten übereinstimmen. Entsprechend der Angaben im

entsprechenden Testkopf der Listings haben die DATA in der Reihenfolge ihrer Ausschrift folgende Bedeutung:

- erwarteter VBUS-Kanal (Sollwert)
- empfangener VBUS-Kanal (Istwert)
  Diese beiden Ausschriften zeigen, in welcher der drei 32-Bit-Gruppen ein Paritätsfehler aufgetreten ist:
  102x = Bits 00...31, 101x = 32...63, 100x = 64...95.
- Nr. des Bits innerhalb einer 32-Bitgruppe (1 = Bit 0)
- Nr. der 32-Bit-Gruppe, hier z.B. 1 = Bits 32...63

Es erscheint hier z.B. zur Fehlereingrenzung sinnvoll, durch Eingabe des LOOP-Kommandos und Eingabe von "SE ST IN" eine Einzelbefehlsabarbeitung in einer kurzen Testschleife, d.h. zwischen der vorangegangenen ERLOOP- und der IFERROR-Anweisung anzuweisen. Durch Eingabe von "SPACE" wird jeweils immer ein Befehl abgearbeitet und der neue PC- Stand ausgegeben. Bei TPC=001316 wird der Sychronisationspunkt "SYNC4C" erreicht, an dem der Operator die Datenbits auf dem CSBUS messen und das fehlerhafte Bit ermitteln kann. An diesem Punkt wird das Mikrowort gerade aus dem WCS gelesen und liegt auf dem CSBUS.

Durch Eingabe irgendeines anderen Zeichens, außer "SPACE" kann man dann wieder aus dem Schrittbetrieb heraus und in den Kommandoeingabemodus gelangen. Löscht man nun z.B. das Fehlerhaltflag HI, setzt das Flag für ein akustisches Signal BE bei Fehlererkennung und setzt das Flag zur Unterdrückung von Fehlerausschriften NE, so wird nach Eingabe von CO (continue) diese Fehlerschleife sehr schnell solange durchlaufen (auch wenn der Fehler beseitigt ist) bis sie durch Eingabe von CTRL/C unterbrochen wird. Diese Methode kann sinnvoll angewendet werden, wenn relativ einfache Fehler, wie z.B. Kontaktunsicherheiten lokalisiert und beseitigt werden sollen. Solange das akustische Dauersignal ertönt, ist der Fehler vorhanden, erlischt es, ist der Fehler beseitigt (im provozierten Fehlerbeispiel durch Stecken des WCS).

4.6.5.3. Mikrotest, Statische Messungen

Es werden Fehler durch Ausschalten der Stromversorgung des Hauptspeichers (vor Einschalten der Anlage !!!!) provoziert.
Als Beispiel sei folgende Fehlerausschrift angenommen:

```
        MIC> DI/TE:163  <CR>
             ----------------
        MTEST1 -V1.0
        3B,
        ?ERROR:109E  TEST:163   SUBTEST:1
        DATA:   00000002        (erwartete Branchbits <2:0>)
                00000006        (empfangene Branchbits <2:0>)
                00000000        (keine Bedeutung)
                00000000        - " -
        TRACE: 00,
        FAILING MODULES: 8915
        MIC> SE SOMM:109B       <CR>    (Unterbrechungspunkt setzen)
             ----------------------
        MIC> LO         <CR>    (Fehlerschleife)
             ----------------
        UPC = 109B      (Unterbrechungspunkt erreicht)
        MIC>SE ST BU    <CR>    (Einzelbuszyklus)
             -------------------
        UPC = 109C      <SP>    (CALL.I/TSTBRE Z.2921 <---------
                        -----   Unterprogrammsprung)            |
        UPC = 1372      <SP>    (Unterprog.TSTBRE Z.2949)       |
                        -----                                   |
        UPC = 1016      <SP>    (RETURN1 , Z.2956)              |
                        -----                                   |
        UPC = 109D      <SP>    (Fehlerschleife, Z.2922)        |
                        -----                                   |
        UPC = 136F      <SP>    (       - " -    Z.2923)        |
                        ----                                    |
        UPC = 109E      <SP>    (       - " -    Z.2924)        |
                        ----                                    |
        UPC = 1184      <SP>    (       usw.           )        |
                        ----                                    |
        UPC = 1185      <SP>                                    |
                        ----                                    |
        UPC = 105A      <SP>                                    |
                        ----                                    |
        UPC = 109B      <CR>    >-----------------------------
                        ----
MIC>SE ST ST    <CR>    (Einzeltaktabarbeitung)
     ----------------
        CPT1    <SP>
                ----
        CPT2    <SP>
                ----
        CPT3    <SP>
                ----
        UPC = 109C      <SP>
                        ----
        CPT1    <SP>
                ----
        CPT2    <SP>
                ----
```

```
        CPT3    <SP>
                ----
        UPC = 1372      <CR>
                        ----
MIC>CL SOMM:109B        <CR>    (Triggerimpulserzeugung
                                bei UPC=109B)
    ------------------------
MIC>LO  <CR>    <CTRL/C>
    --------------------
MIC>
```

Der Fehler-PC ist vierstellig und verweist demzufolge auf einen Mikrotest, und zwar Test-NR.: 163 in der Sektion 3B. In den Listings beginnt dieser Test auf der Zeile 2882 und der Fehler-UPC ist auf Zeile 2924 zu finden. Dieser Test prüft die Branchbits <2:0> während eines Interrupts ab. Im Subtest 1 wird erwartet, daß Branchbit 1 gesetzt ist und die Branchbits 0 und 2 gelöscht sind.

Um den Fehler auf der Steckeinheit durch statische Messungen, bzw. Abfrage von Registern oder VBUS durch EX-Kommandos lokalisieren zu können, ist es zweckmäßig, zum Einzelbuszyklus überzugehen. Verfolgt man in den Listings den Mikrocode rückwärts, so zeigt sich eine mögliche Schleife zwischen "ERROR2" (Zeile 2924) und "ERLOOP" (Zeile 2919). Es wird ein sinnvoller Punkt in dieser Schleife zum Stop des Mikrotestes gewählt, in diesem Fall z.B. beim UPC=109B (Zeile 2920). Es wird durch Eingabe von "SE SOMM:109B" ein Unterbrechungspunkt gesetzt und durch Eingabe des "LOOP"-Kommandos die Testschleife gestartet, wobei das HI- bzw. HD-Flag gelöscht sowie das NE- und das LO-Flag gesetzt werden und ein CO-Kommando ausgeführt wird. Beim Erreichen des Unterbrechungspunktes UPC=109B geht der Mikrotest in den Halt und die Mikrodiagnose in den Kommandoeingabemodus. Es ist jetzt zur schrittweisen Abarbeitung das Kommando "SE ST BU" (Einzelbuszyklus) überzugehen. Nach jedem Zyklus (jeweils Eingabe von "SPACE") können dann die relevanten Signale gemessen werden.

Der UPC = 109B ist der 1. Mikrobefehl nach der ERLOOP-Anweisung, auf den nach Durchlaufen der Fehlerschleife zurückgesprungen wird (nach ERROR2). Der nächste UPC ist 109C (Z.2921). Er beinhaltet einen Sprung in ein testinternes Unterprogramm CALL.I/TSTBRE nach UPC = 1372 (Z. 2921), wo die zu prüfende Funktion abgetestet wird. Vom UPC = 1016 (Z. 2956) erfolgt dann ein Rücksprung in die Fehlerschleife nach UPC=109D, 136F, 109E. Von hier aus erfolgt dann ein Unterprogrammsprung in die sektionsinterne Fehlerbehandlungsroutine ERROR2 (UPC=1184, 1185, 105A; Zeile 1922 bis 1924). Da kein CALLCON (Kommandoeingabemodus) angewiesen ist, erfolgt Rücksprung in die Fehlerschleife nach UPC=109B usw.

Durch Eingabe irgendeines anderen Zeichens außer "SPACE" kann der Einzelschrittmodus beendet werden. Die vorher gesetzten Programmsteuerflags werden gelöscht und das "HI"-Flag gesetzt, um die normalen Fehlererkennungsbedingungen wiederherzustellen. Es können so auch nach jedem Mikrobefehlsschritt durch <CR> und entsprechende EX-Kommandos in Abhängigkeit von der zu testenden Funktionsgruppe und von dem zuletzt ausgeführten Mikrobefehl bestimmte Registerinhalte bzw. der entsprechende VBUS-Kanal abgefragt und ggf. durch DE-Kommandos Register definiert mit anderen Informationen beschrieben werden. Eine "CL SOMM"-Eingabe löscht das Microbreakpointregister (MBRK) und den entsprechenden Stop. Durch Eingabe von "CO" kann die Diagnose dann mit dem nächsten Test fortgesetzt oder durch "DI" wieder von Anfang an abgearbeitet werden. Bei entsprechender Erfordernis kann dieses Verfahren anstelle von Einzelbuszyklusbetrieb im Einzeltaktbetrieb durch Eingabe von "SE ST ST" anstatt "SE ST BU" angewendet werden.

4.6.5.4. Mikrotest, Dynamische Messungen

Es wird die gleiche Fehlerausschrift wie bei Pkt. 5.2 angenommen. Sind zur Fehlereingrenzung dynamische Messungen mittels Oszillograf oder Analyzer erforderlich, so ist die Vorgehensweise prinzipiell die gleiche, nur daß kein Schrittbetrieb angewiesen wird und anstelle von "SE SOMM:109B" "CL SOMM:109B" beispielsweise einzugeben ist. Es würde dann anstelle eines Stops beim Erreichen dieses UPC's ein Synchronimpuls zur Triggerung des Meßgerätes erzeugt werden. Die Testschleife wird unter jedesmaligem Erzeugen des Synchronimpulses solange durchlaufen, bis durch Eingabe von "CTRL/C" ein Operatorinterrupt erfolgt und dadurch die Mikrodiagnose in den Kommandoeingabemodus geht.

Das Triggersignal kann am oberen Meßpunkt, nahe der Griffseite des Mikrosequenzers USC, 8923, Steckplatz 23 abgenommen werden. Welcher UPC in der CL SOMM:xxxx anzugeben ist, ergibt sich aus der Überlegung, bei welchem Mikrobefehl die gewünschten, zu messenden Signale auftreten bzw. am günstigsten zur Fehlerlokalisierung erfaßt werden können.

4.7. Abkürzungsverzeichnis

In diesem Kapitel werden folgende Abkürzungen verwendet:

CPU	Zentrale Verarbeitungseinheit
DI	Directory-Programm
DM	Diagnosemonitor
FM	Mikrotestfehlermonitor
FSE	Foliensperichereinheit
HM	Hardcoretestmonitor
HR	Hauptrechner (K 1840)
HT	Hardcoretest
KP	Kommandoeingabeprogramm
KSS	Konsolsubsystem
MEM	Hauptspeicher
MF	Mikrotestfehlerprogramm
MM	Mikrotestmonitor
MT	Mikrotest
PBT	Prüfbit

4.8. Auflistung der Tests

Spalte 1 = Sektions-Nr., Spalte 2 = Test-Nr.
Spalte 3 = Testinhalt

4.8.1. Diskette DS2

4.8.1.1. Hardcoretests

01	001	Konsol-Adapter-Register-Antwort
	002	Konsol-"ZUM ID"-Register Datenerhalt
	003	Konsol-"MCR"-Register Datenerhalt
	004	VBUS-Selbsttest
	005	Konsol-"IDCS"-Register Datenerhalt
	006	Konsol-RXDNE und -TXRDY-Register Datenerhalt
02	007	TXRDY- und -RXDNE-Interrupts
	008	IDBUS-Datenleitungen Datenerhalt
	009	Steuerung des Konsol-Taktgenerators
	00A	Funktion des Konsol-IDBUS-Zyklus
	00B	Konsole vom ID-Register, Taktsteuerung und Datenerhalt
03	00C	Konsol-Maintenance-Return
	00D	RXCS-Register von der IDBUS-Seite
	00E	TXCS-Register zum IDBUS
	00F	IDBUS-Register Adreßerhalt
	010	CIB-Initialisierungsfunktion
04	011	Konsol-Register Doppeladressierung
	012	Lesen WCS-Datenregister
	013	Initialisierung des Befehlsspeichers
	014	WCS-Adreßregister Datenerhalt
	015	WCS-Adreßregister Zählerlogik
	016	Mikrostack Datenerhalt
	017	Mikrostack Doppeladressierung
	018	Maintenance-Return Datenerhalt
05	019	Maintenance-Return-Microstack-Increment
	01A	Mikrostack Schreiberlaubnis
	01B	WCS-Paritätsgenerator
	01C	CSBUS Datenerhalt
	01D	PCS Paritätsprüfung
06	01E	WCS Doppeladressierung 5.,6.,7. & 8.K
07	01F	WCS dynamischer Speichertest
08	020	NUA-Bus-Treiber
	021	UBEN Felddekodierung
	022	USUB-Feld "CALL"-Funktion
	023	USUB-Feld "Return"
	024	USUB-Feld Auswahlschalter
	025	UJMP-Feld Datenerhalt

09	026	Mikrobreak-Register Datenerhalt
	027	Mikrobreak-Komparator
	028	Mikro-ECO
	029	Retten des Mikrostack mit CS-Paritätsfehler
0A	02A	Initialisierung Mikro-Trap
0B	02B	IDBUS zum Datenkanal-Interface
	02C	ALU ALEG Datenkanäle "RAMX Q"
0C	02D	ALU ALEG Datenkanäle "RAMX D"
	02E	Datenkanäle ALU BLEG "RBMX Q"
0D	02F	Datenkanäle ALU BLEG "RBMX D"
0E	030	Datenkanäle ALU Arithmetik-Modus
0F	031	Datenkanäle ALU Arithmetik-Modus
10	032	ALU Carry- und Modus-Steuerung
	033	Datenkanäle KMX
	034	IDBUS, Schreiben / Lesen
11	035	"D bytes" Branch
12	036	ALU Branch
13	037	Scratch-Pad Datenerhalt
14	038	Scratch-Pad Doppeladressierung
15	039	ID-Maint-Lockout während CLK läuft
	03A	RUN-Bit im MCS-Register
	03B	Konsol-ACK-Bit im MCS-Register
	03C	Konsol-Komando-Modus-Bit im MCS-Register
	03D	Accelerator VBUS-Integrität
16	03E	reserviert
17	03F	reserviert
18		reserviert
19		reserviert
1A		reserviert
1B		reserviert
1C		reserviert
1D		reserviert
1E		reserviert
1F		reserviert

4.8.1.2. Mikrotests 1

20	040	verschiedene D-Register Funktionstests
	041	D Linksverschiebung
	042	D Rechtsverschiebung
	043	BEN/D3-0-Test
	044	BEN/D,NE,0-Test
	045	verschiedene Q-Register Funktionstests
	046	Q-Register-Test, Linksverschiebung
	047	Q-Register-Test, Rechtsverschiebung
	048	durchfaufende "1" im RA & RB Register
	049	durchlaufende "0" im RA & RB Register
	04A	RA & RB Registers vorwärts Doppeladressierung
	04B	RA & RB Registers rückwärts Doppeladressierung
	04C	durchlaufende 1'en in RA & RB Register-Tests
	04D	durchlaufende 0'en in RA & RB Register-Tests
21	04E	durchlaufende "1" im RC Register-Test
	04F	durchlaufende "0" im RC Register-Test
	050	RC Register vorwärts Doppeladressierung
	051	RC Register rückwärts Doppeladressierung
	052	gallop 1's in RC Register-Test
	053	gallop 0's in RC Register-Test
	054	Nibble swab D , gleitende 1 durch Nullen
	055	Nibble swab D , gleitende 0 durch Einsen
	056	Test USI, Shiftereingang nutzt Q-Register
	057	Test USI, Shiftereingang nutzt D-Register
	058	AMX-Test
22	059	ALU-Test
	05A	PSL Register Datenerhalt
	05B	ALU/A+B+PSL.C-Test
	05C	SK ROM Prüfsummentest
	05D	D MX und SHF MUX-Test
23	05E	SC and KMX/SC-Test
	05F	SC BEN-Test A
	060	SC BEN-Test B
	061	SC BEN-Test C
	062	Masken-Test
	063	SMX/ALU,EXP-Test
	064	Dual Shifter-Test
	065	EALU/A+1, EAMX/SC & SMX/EALU-Test
	066	EALU/A-Test
	067	EBMX/KMX and EALU/B-Test
	068	FE and SMX/FE-Test
	069	EBMX/FE-Test
	06A	EBMX/AMX, EXP-Test
	06B	Statusregister und EAMX/STATE-Test
	06C	EALU-Test
	06D	EALU/NABS, A-B-Test
	06F	EALU/NABS, A-B ROM-Tes

24	06F	QK dezimal Konstantentest 1
	070	QK dezimal Konstantentest 2
	071	D Laden oder Verschieben Bedingungsfunktion
	072	Verschiebewerteingang im DAL
	073	Verschiebewertlogik und EBMX/SHF-Wert
	074	USI-Test unter Verwendung des SHF MUX
	075	Vorzeichenkontrolle und BEN-Vorzeichen-Test
	076	BMX /PACKED.FL Test
	077	BEN ALU.N Test
	078	Schreiben RAB mit DT/BYTE
	079	Schreiben RAB mit DT/WORT
	07A	SC AS Adreßquelle für RC-Register
	07B	RA und RB-Adressierung durch SC
	07C	LA-RA[0-7] - Test
25	07D	LAB R[1] und Schreiben RC Funktionstest
	07E	Schreiben R[1] und Latch RC-Funktionstest
	07F	LA- und LB-Steuerlogik
	080	LC-Steuerlogik
	081	RA- und RB-Schreibsteuerlogik
	082	RC-Schreibsteuerlogik
	083	VA, PC und BMX/PC
	084	VA-Inkrementfunktionstest
	085	Inkrement mit 1 im PC
	086	Inkrement mit 2 im PC
	087	Inkrement mit 4 im PC
	088	VIBA-Test
	089	Test des PCSV-Registers
	08A	BEN RLOG Stack-Empty-Test
	08B	RLOG-Stack-Test
	08C	BEN EALU.N-Test
	08D	BEN EALU.Z-Test
	08E	BEN ALU <01:00> -Test
	08F	UTRAP bei ungültigem Gleitkommaoperanden
26	090	BEN STATE <03:00>
	091	BEN STATE <07:04>
	092	BEN SS (Vorzeichenquelle)
	093	BEN LA <01:00>
	094	BEN D <01:00>
	095	BEN / OF Test1
	096	BEN / OF Test2
	097	BEN PTE Gültigkeitstest
	098	BEN PSL <27>, FPD-Test
	099	BEN / 1C - Test
	09A	BEN/NEST.ERR und CES<NEST.ERR>
	09B	BEN PSL.CC - Test
	09C	BEN / 1C , Konsolmodus
	09D	BEN / 1C , Kernelmodus
	09E	BRANCH ENABLE 3 -- ALU C-Bit
	09F	BRANCH ENABLE 1B -- ALU Z-Bit
	0A0	IDBUS-Adressierung durch SC
27	0A1	VAMX/VA, PAMX/VAMX, PABUS und Timeout-Adreß-Register
	0A2	VAMX Kompatibilitätsmodus
	0A3	MSC/INHIBIT Kompatibilitätsmodus unter Verwendung VAMX
	0A4	TB-Schreibimpulslogik
	0A5	Datenkanäle zum und vom Datenspeicher

28	0A6	Adreßkanäle zum und vom TAG-Speicher
	0A7	Komparator Gruppe 0
	0A8	Komparator Gruppe 1
	0A9	TB-Paritätslogik
	0AA	UTRAP-Bitmodifizierung
	0AB	TB gültiges Bit UTRAP
29	0AC	ALIGN UTRAP Logik
	0AD	CROSS PAGE UTRAP unter Verwendung von DI/LONG
	0AE	CROSS PAGE UTRAP unter Verwendung von DI/WORD
	0AF	CROSS PAGE UTRAP ENABLE
	0B0	TB-Schutzlogik
	0B1	IPA-Registertest
	0B2	READ.V.NEWPC, HIT IN GROUP 0
2A	0B3	READ.V.NEWPC, HIT IN GROUP 1
	0B4	TB-Gruppe 0 PTE , statischer Speichertest
	0B5	TB-Gruppe 1 PTE , - " -
	0B6	TB-Gruppe 0 TAG , - " -
	0B7	TB-Gruppe 1 TAG , - " -
	0B8	Doppeladressierungstest der TB-Adreßmatrix
	0B9	- " - der TB-Datenmatrix
	0BA	dynamischer Test des TB-Adreß-Speichers
	0BB	- " - -"- des TB-Daten-Speichers (PTE)
2B	0BC	BRANCH ENABLE 11 --- LAST REFERENCE
	0BD	TB-Fehlerregister 0, Bits <15:10>
2C	0BE	UTRAP bei ungerader Adresse, Test 1
	0BF	- " - , Test 2
	0C0	TB MISS UTRAPS, Test 1
	0C1	- " - , Test 2
	0C2	UTRAPS bei Zugriffsverletzung, Test 1
	0C3	- " - , Test 2
	0C4	- " - , Test 3
	0C5	- " - , Test 4
	0C6	M-Bit-UTRAPS, Test 1
	0C7	- " - , Test 2
	0C8	ALIGNMENT-UTRAPS , Test 1
	0C9	- " - , Test 2
	0CA	CROSS-PAGE-UTRAPS, Test 1
	0CB	- " - , Test 2
	0CC	TB-PARITY-UTRAPS , Test 1
	0CD	- " - , Test 2
2D	0CE	CACHE , Schreibimpulslogik
	0CF	Adreßbit 2 in Gruppe 0
	0D0	Adreßbit 2 in Gruppe 1
	0D1	Datenkanäle zur Gruppe 0
	0D2	- " - Gruppe 1
	0D3	Adreßkanäle TAG-Speicher , Gruppe 0
	0D4	- " - , Gruppe 1
	0D5	Gruppe 0 , HIT/MISS-Komparator

2E	0D6	Gruppe 1 , HIT/MISS-Komparator
	0D7	Adreßparitätslogik
	0D8	Datenparitätslogik
	0D9	FORCE-MISS-Bit und Paritätsfehler , Gruppe 0
	0DA	- " - , Gruppe 1
	0DB	PA <29> - Wirkung auf HIT/MISS-Komparator
	0DC	statischer Datenspeichertest der Gruppen 0 und 1
	0DD	statischer Adreßspeichertest der Gruppe 0
	0DE	- " - Gruppe 1
2F	0DF	Doppeladressierungstest des CACHE TAG STORE
	0E0	- " - des CACHE VALID BIT
	0E1	- " - der CACHE-Datenmatrix
	0E2	CACHE-Paritäts-Mikro-Trap durch MCT-Funktion
	0E3	MOVING INVERSIONS in Datenspeichern der Gruppe 0
	0E4	- " - Gruppe 1
30	0E5	MOVING INVERSIONS in Adreßspeichern der Gruppe 0
	0E6	- " - Gruppe 1
	0E7	MDAL Test 1
	0E8	MDAL Test 2
	0E9	MDAL Test 3
	0EA	MDAL Test 4
	0EB	BAL Test 1
31	0EC	BAL Test 2
	0ED	BAL Test 3
	0EE	BAL Test 4
	0EF	BAL Test 5
32	0F0	VIBA, Inkrementfunktion
	0F1	IPA , - " -
	0F2	AUTO-RELOAD-Test 1
	0F3	- " - 2
	0F4	- " - 3
	0F5	- " - 4
	0F6	- " - 5
33	110	Befehlspuffer
	111	Verzweigungstest,Op-Code 00 bis 05 und FF
	112	Steuerung der Verschiebung Byte 0
	113	- " - Byte 1
	114	- " - Byte 2 bis 6
	115	Datenerhalt bei Byte-Rotation
34	118	Datenerhalt bei Verschiebung Multiplexer
	119	Erzeugung gültiges Bit
35	11C	Fortsetzung Test 119
	11D	STALL-Steuerlogik
	11E	Dekodierung Op-Code "A FORK"
	11F	- " - Branch-Op-Codes "A FORK"
	120	Auswahlschalterlogik
	121	Dekodierung Op-Code "B FORK"
	122	- " - "C FORK"
	123	D-MUX-Auswahl und Datenerhalt
	124	Dekodierung BDEST und 16-Bit-BDEST

```
36   127   EXECUTION POINT COUNTER und FIRST PART DONE
     128   Dekodierung Kompatibilitätsmodus A FORK
     129      - " -        - " -            A FORK -Branch
     12A      - " -        - " -            B FORK
     12B      - " -        - " -            C FORK
     12C   Kompatibilitätsmodus STALL-Logik
     12D           - " -            D-MUX - Auswahl
     12E   Steuerung IBA-Bit <1:0>
     12F   8908 - IRD - ROM - Branch-Test
     130   8908 - IRD  EXECUTION POINT COUNTER

37   131   IB-Request und Zähler
     132   K 1800 - Modus, Spezifikator ist Konstante
     133   Kompatibilitätsmodus SP1 und SP2 konstant
     134   Inkrement PC-Register durch "N"
     135   befehlsabhängige SCRATCH-PAD-Adressierung
     136   BMUX erhält PC- oder LB-Funktion
     137   SHIFTER-Auswahl durch befehlsabhängigen Datentyp
     138   Kompatibilitätsmodus CONTEXT LATCH

38   13D   Datenkanäle CONTEXT LATCH
     13E   quellenabhängiges Vorzeichen in IR<1>
     13F   ALU-ROM , befehlsabhängig
     140   Bedingungscode LOAD=1,2,4,3&5&6(nur C- und V-Bit)
     141           - " -  LOAD=3 (N- und Z-Bit)
     142           - " -  LOAD=5&6 (N- und Z-Bit)
     143           - " -  LOAD (C-Bit), befehlsabhängig
     144           - " -  LOAD (V-Bit),     - " -

39   146   Bedingungscode LOAD (Z-Bit), befehlsabhängig
     147           - " -  LOAD (N-Bit),     - " -
     148   CM      - " -  LOAD=4
     149   CM      - " -  LOAD=5 (Z-Bit)
     14A   CM      - " -  LOAD=6 (C-,Z- und N-Bits)
     14B   CM      - " -  LOAD=6 (V-Bit)
     14C   CM      - " -  LOAD,  befehlsabhängig
     14D   CM BRANCH ENABLE 8....Ende DP1
     14E           - " -    9....PC Modus
     14F           - " -    A....RE1
     150   K 1800-Modus BRANCH ENABLE 8....Datentyp

3A   153   K 1800-Modus BRANCH ENABLE 9....IR<2:1>
     154           - " -            B....IB
     155   MEMORY REFERENCE CANCELS
     156   "D MD" verboten durch "ABORT CYCLE"
     157   CES-Register-Arithmetik-Trap-Codes (Bit <6:4>)
     158   Löschen PSL durch Interrupt und EXEPT.ACKNOWL.
     159   Bits "vorheriger Modus" ausgewählt de "CHK CHG MOD"
     15A   Bits "aktueller Modus" und Prüfung Modusänderung
     15B   Bits "aktueller und vorheriger  Modus" laden während EXEPT.ACK.
     15C   PSL "IS"-Bit laden während INTERR.ACK.
     15D   PSL (TP)-Bit, trap-abhängig
     15E   PSL-Bits, die eine Funktion von "LOAD FPD" sind
     15F   Datenerhalt im Vektorregister Bit <31:16>
     160   Software-Interrupt-Anforderungs-Register
```

3B	161	Intervallzähler
	162	IPL-Aktivbits im SIR-Register U. IPL-IPL-aktiv im PSL
	163	Branch-enable E....Interrupt
	164	Interruptanforderungs-Komparator
	165	Erzeugung des Befehlspuffer-Service-Bits
	166	HALT-abhängiges Bit
	167	Inkrement des PC-Registers durch "N" wenn "SAVE" aktiv
3C	168	Datenerhalt CES-Register Bit <11:00> vom IDBUS
	169	CES-Register Bits <11,10,8,7> und Bedingungs-Code LOAD=1
	16A	CES-Register ALU N Bit
	16B	CES-Register Bits <15:12> , CS-Paritätsfehler
	16C	Empfänger- und Intervallzeitgeber-Interrupts
	16D	Konsol-Interrupt-Disable
	16E	IDBUS-"ATEMP"-Register
	16F	IDBUS-"BTEMP"-Register

4.8.2. Diskette DS3 , Mikrotest 2

3D	171	BEN A , aktuellueller Modus kleiner als AST-Level
	172	Schreibtest Befehlspuffer
	173	Retten Kontext und Datentyp Quadword
	174	ALU Z Branch mit Datentyp Quadword
	175	Lesen NEWPC mit TB-Miss
	176	Lesen NEWPC mit Zugriffsverletzung
	177	Lesen NEWPC mit TB-Paritätsfehler
	178	reserviert
	179	reserviert
	17A	reserviert
3E	17B	ISB-Hold-Test
	17C	Timeout-Test
	17D	CPU-ID-Test
	17E	Test Befehlsadresse
	17F	Antwort ISB-Nexus
	180	mindestens ein Speicherkontroller vorhanden
	181	Timeout-Zähler
	182	reserviert
	183	reserviert
	184	reserviert
3F	185	erzwungenes Timeout
	186	Zugriff Konfigurationsregister
	187	Test Daten-Latch 1 und 2
	188	Test Startadresse
	189	Kontroller-Latch
	18A	Speicher Schreiben / Lesen
	18B	reserviert
	18C	reserviert
40	18D	erzwungener ISB-Fault
	18E	mehrfacher CPU und Speicher XMIT-Fault
	18F	reserviert
	190	reserviert
41	191	Datentest ID-Register 1C
	192	Comparator-Register Zählfeld
	193	SILO sperren durch ID-Vergleich
	194	SILO sperren durch Vergleich ID und TAG
	195	SILO sperren durch Vergleich ID, TAG und Funktion
	196	reserviert
	197	reserviert
	198	reserviert
42	199	SILO-Test
	19A	prüfen ISB-Kommando und STALL durch SILO-Test
	19B	reserviert
	19C	reserviert
43	19D	wie Test 19A
	19E	reserviert
	19F	reserviert
	1A0	reserviert

44	1A1	SILO und ISB-Interface
	1A2	ISB-Disable
	1A3	reserviert
	1A4	reserviert
45	1A5	Paritätsfehler-Mikrosequenzer
	1A6	Paritätsprüfung und -generierung IDBUS
	1A7	Lese-/Schreib-Zugriff ROM
	1A8	Test des byteweisen Schreibens in den Speicher
	1A9	ungezieltes Schreiben/Lesen in bzw. aus dem Speicher
	1AA	reserviert
	1AB	reserviert
	1AC	reserviert
46	1AD	single state ISB-Interface
	1AE	single time state ISB-Interface
	1AF	single bus state ISB-Interface
	1B0	single bus state ISB-Interface
	1B1	reserviert
	1B2	reserviert
47	1B3	ISB, Funktionsprüfung virtueller Operator
	1B4	Test Speichergröße im "Interleave"-Mode
	1B5	reserviert
	1B6	reserviert
48	1B7	Auswahl Kontroller
	1B8	reserviert
49	1B9	ausgewählte Adressierung BOARD/BANK
	1BA	reserviert
4A	1BB	ausgewählte Adressierung Spalte/Zeile
	1BC	reserviert
4B	1BD	Erzeugung Check-Bits
	1BE	Erkennung Einzelbitfehler
	1BF	Korrektur Einzelbitfehler, Teil 1
	1C0	" " " 2
	1C1	" " " 3
	1C2	Doppelbitfehler Teil 1
	1C3	" " 2
	1C4	reserviert
	1C5	reserviert
	1C6	reserviert
4C	1C7	CRD inhibit
	1C8	"Error Log"- und "High Error Rate"-Bits-Test
	1C9	"Error Logging" (CRD/RDS)-Test
	1CA	Fehleradreßregistertest
	1CB	Einzel/Doppel-Bitfehler-Erkennung bei erweitertem Lesen
	1CC	reserviert
	1CD	reserviert
	1CE	reserviert

4D	1CF	vorläufiger CACHE-Test
	1D0	Quad-Wort-Placement im CACHE bei Bytelesen
	1D1	vorläufiger Datentest CACHE/Speicher
	1D2	Aktualisierung CACHE durch Schreiben Hits
	1D3	reserviert
	1D4	reserviert
4E	1D5	CACHE/ISB ungültig
	1D6	Disable Interrupt
	1D7	reserviert
	1D8	reserviert
4F	1D9	Cancel ISB-Zyklen
	1DA	reserviert
50	1DB	Lesen Befehlspuffer falsch
	1DC	Datenparitätsfehler Befehlspuffer/CACHE
	1DD	Befehlspuffer-Timeout
	1DE	Lesen Befehlspuffer mit ungültiger Adresse
	1DF	Paritätsfehler, Lesen Befehlspuffer und CACHE
	1E0	ID und CP lesen
	1E1	Befehlspuffer und CP lesen
	1E2	" lesen und CP schreiben
	1E3	" und CP ungültig
	1E4	" und RDS
	1E5	Löschen ISB-Fehlerregister
	1E6	Timeout-Adreß-Register-Bit < 31:28 >
	1E7	reserviert
	1E8	reserviert
51	1E9	Auto-Refill-Hit und CP-Read-Hit
	1EA	" " -Miss und CP- " -Miss
	1EB	" " und ROM-Enable
	1EC	" " und CPU schreiben
	1ED	" " und CP ungültig
	1EE	" " mit ISB-Disabled und ISR
	1EF	" " und IB flush
	1F0	STALL<=TO TRB Enable*BUFF FULL*DisableISB*-AR der ISB
	1F1	STALL während lesen Interrupt-Summary
	1F2	Schreiben, CPU-Miss und Daten-Speicher-Paritätsfehler
	1F3	" " -Hit und CACHE-Daten-Paritätsfehler
	1F4	Maske bei Befehlspuffer lesen
	1F5	Auto-Refill-Canceled durch Befehlspuffer flush
	1F6	reserviert
52	1F7	Einzelbitfehler bei maskiertem Schreiben
	1F8	Doppelbitfehler " " "
	1F9	Datenerhalt bei "Quick Verify Array"-Integitätslesen
	1FA	reserviert
53	1FB	verschiedene ISB/CACHE - Tests
	1FC	ROM-Prüfsumme
	1FD	reserviert
	1FE	reserviert

54	1FF	erweitertes maskiertes Schreiben
	200	CRD/RDS-Fehler bei erweitertem maskiertem Schreiben
	201	Lesen Langwort
	202	CRD/RDS-Fehler bei Lesen Langwort
	203	reserviert
	204	reserviert
55	205	ISB ungültig bei Adreß-Paritätsfehler Gruppe 0
	206	" " " " " " 1
	207	CPU Lesen dabei ISB ungültig
	208	ISB-Anforderungen 4,5,6 oder 7
	209	reserviert
	20A	reserviert
56	20B	ISB ungültig
	20C	ISB ungültig bei Schreiben MISS-Paritätsfehler in Gruppe 0
	20D	" " " " " " 1
	20E	I/O-Schreib-Zyklus in Gruppe 0, ISB nicht ungültig
	20F	" " " " " 1 " " "
	210	Lesetest Interrupt-Summary
	211	reserviert
	212	reserviert
57	213	CRD/RDS-NEXUS-Test
	214	reserviert
	215	reserviert
	216	reserviert
58	217	ISB-NEXUS-FAULT-Test
	218	reserviert
59	219	Cleanup-Code für Speichertest
	21A	reserviert
	21B	reserviert
	21C	reserviert
5A	21D	ID-Bit FPA-Statusregister
	21E	Enable-Bit FPA-Statusregister
	21F	Maintenance-Trap
	220	NAD-Feld FPA-Mikrosequenzer
	221	Test 1 Account-Control-Trap
	222	" 2 " "
	223	Microbreak-Register und SYNC
	224	FPA im IRD-Status
	225	FPA/CPU-SYNC , wait hand shake
	226	Befehlstest FPA
	227	BEN/1
	228	A-FORK
	229	B-FORK
	22A	Datenwege FPA IDBUS/ACCBUS
	22B	FPA Bus A
	22C	reserviert

5B	22D	FAD AR, BR-Latch Test 1
	22E	" " " " " 2
	22F	FPA Scratch-Pad-Copy Daten Test 1
	230	" " " " " " 2
	231	Adressierungtest für FPA-Scratch-Pad-Copy
	232	Adressierung allgemeine Register SP1- RA
	233	" " " SP2 - RB
	234	" " " SP2+1 - RA
	235	" " " PRN+1 - RB
	236	" " " PRN+1 - RA
	237	reserviert
	238	reserviert
5C	239	Adressierung allgemeines Register PRN - RB
	23A	CPSYNC/WAIT, Löschen Mikrowort
	23B	Literal-Register
	23C	Scratch-Pad-Daten
	23D	" " Konflikt-Adresse
	23E	Vorzeichen-Latch SA u. SB
	23F	reserviert
	240	reserviert
	241	reserviert
	242	reserviert
	243	reserviert
	244	reserviert
	245	reserviert
	246	reserviert
	247	reserviert
	248	reserviert
5D	249	BEN/4 Bit 2 , poly done
	24A	BEN/4 Bit 1 , CPU SYNC
	24B	BEN/4 Bit 0 , float Op-Code
	24C	LA-Register , AMX/LA u. EALU/A
	24D	LB-Register , AMX/LB u. EALU/A
	24E	BMX/LB u. EALU/A+B mit A=0
	24F	BMX/80
	250	XR-Register u. BMX/XR
	251	PR-Register, Bit 00...07 u. AMX/PR, Bit 00...07
	252	EALU/K3FF
	253	Laden LA
	254	" LB
	255	" PR
	256	" XR
	257	PR<09:08>, EALU<09:08> u. Laden ACC.CC Test 1
	258	" " " " " " " " " 2
	259	" " " " " " " " " 3
	25A	" " " " " " " " " 4
	25B	" " " " " " " " " 5
	25C	BEN/7, Test 1
	25D	" " 2
	25E	" " 3
	25F	" " 4
	260	" " 5
	261	reserviert
	262	reserviert

5E	263	EALU/A+B
	264	EALU/A-B
	265	BEN/3 und BEN/5 Bit 2
	266	ASHF u. EXP.DIFF Test 1
	267	" " " " " 2
	268	" " " " " 3
	269	" " " " 4
	26A	" " " " 5
	26B	" " " " 6
	26C	
5F	26D	AMXC/COND
	26E	BEN/5 Bit 0 Test 1
	26F	" " 0 " 2
	270	" " 1 " 1
	271	" " 1 " 2
	272	EALU/A+B Test 1
	273	" " 2
	274	EALU/A-B Test 1
	275	" " 2
	276	reserviert
60	277	Komplement AR
	278	" BR
	279	FPA-Fehler-Latch und CPU BEN/6
	27A	reserviert
	27B	reserviert
	27C	reserviert
61	27D	Vorzeichenlogik für MUL/DIF
	27E	" " EXACT ZERO,UFLO,OVFLO
	27F	" " ADD/SUB mit LA>LB
	280	reserviert
62	281	Vorzeichenlogik für ADD/SUB mit LA=LB
	282	" " " " LA<LB
	283	" " SA,XOR,SX
	284	" " POLYADDL und ADD/SUB
	285	reserviert
	286	reserviert
63	287	Normalisierershift innnerhalb eines Bereiches
	288	" ROM-Test 1
	289	" " " 2
	28A	EXP.OVERFL während Rundung
	28B	ACC. OVERRIDE Test 1
	28C	" " " 2
	28D	" " " 3
	28E	" " " 4
	28F	reserviert
	290	reserviert

64	291	verifizierte Divisionszustände float und double
	292	Division double floating 1 durch 0,1
	293	" " 0,1 durch floating 1
	294	" " floating 1 durch floating 1
	295	" " mit ausgewählten Operanden
	296	NALU CARRY PROPAGATE FAULTS
	297	Division float floating 1 durch floating 1
	298	reserviert
65	299	Integer-Multiplikation Produkt LO
	29A	" " " HI
	29B	Float- " floating 1 durch 0,1
	29C	" " 0,1 durch floating 1
	29D	EMODF, floating 1 durch 0,1
	29E	" , 0,1 durch floating 1
	29F	reserviert
	2A0	reserviert
66	2A1	EMODD, floating 1 durch 0,1
	2A2	" , 0,1 durch floating 1
	2A3	Multiplikation double floating 1 durch 0,1
	2A4	" " 0,1 durch floating 1
	2A5	Vergleich Multiplier-ROMS
	2A6	reserviert

4.9. Anhang
Zuordnung Test - Funktionsgruppen

In diesem Anhang wird dargestellt, in welchen Tests, welche Funktionsgruppen bzw. Leiterplatten ggf. als defekt ermittelt werden, d.h. in dem betreffenden Test angesprochen werden. Kann ein aufgetretener Fehler nicht eindeutig einer Funktionsgruppe zugeordnet werden, so werden alle möglicherweise fehlerhaften Funktionsgruppen am Terminal ausgegeben, wobei die Reihenfolge der Fehlerwahrscheinlichkeit entspricht. Ferner kann in einem solchen Fall der Fehler auch an dem Bus, an den diese Funktionsgruppen angeschlossen sind, aufgetreten sein. Dann wird auch der betreffende Bus als mögliche Fehlerursache ausgegeben.

1. Hardcoretests S.1
2. Mikrotest 1 S.3
3. Mikrotest 2 S.9

1. Hardcoretests

```
   FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
   Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
       | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
   ----|-------------------------------------------------------|
   BLP:| 888888888888888888888 8888  8888 88888 8888         |
   \   | 999999999999999999999 7777  5555 99999 8888 VCI     |
    \  | 000000001111111322202 0000  0000 22222 1111 BSD     |
SE/TE\ | 234567890123456123910 1234  1234 45678 3421         |
   ---------------------------------------------------------
01 001 |                    X
   002 |                    X
   003 |                X   X
   004 |                    X                         X
   005 |  X           X     X                           X
   006 |                    X
02 007 |                    X
   008 |  X  XXX      X    XX                           X
   009 |                X   X
   00A |                    X
   00B |                    X                           X
   00C |                   XX
03 00D |                    X
   00E |                    X
   00F |                                                X
   010 |                    X
```

1. Hardcoretests , Fortsetzung

```
      ---------------------------------------------------------------
      FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
      Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
          | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
      ----|-------------------------------------------------------|
      BLP:| 88888888888888888888 8888  8888 88888 8888          |
      \   | 99999999999999999999 7777  5555 99999 8888 VCI      |
       \  | 00000000111111322202 0000  0000 22222 1111 BSD      |
   SE/TE\ | 23456789012345612391 0 1234  1234 45678 3421         |
      ---------------------------------------------------------------

04 011 |                 X
   012 |  Ermittlung und Ausgabe der WCS-Größe am Terminal
   013 |   Initialisierung des WCS
   014 |                X
   015 |                X
   016 | X           X  X
   017 |                X
   018 | X   X       X  X
05 019 |                X
   01A |                X
   01B |              XXX  X
   01C |              X X  X                        X
   01D |               X
06 01E |              X    X
07 01F |               X   X
08 020 |                X
   021 |                X
   022 |                X
   023 |                X
   024 |                X
   025 |                X
09 026 |                X
   027 |             X  X
   028 |                X
   029 |     X       X  X
0A 02A |             X  X
0B 02B |        XXXXX X
   02C |        XXXXX
0C 02D |        XXXXX
   02E |        XXXXX
0D 02F |        XXXXX
0E 030 |        XXXXX
0F 031 |         XXXX
10 032 |            X
   033 |         XXX
   034 |             X    X
11 035 |        X    X   X
12 036 |         XXX XX   X
13 037 |         XXXX
14 038 |         XXXX
15 039 |                  X
   03A |             X    X
   03B |             X    X
   03C |             X    X
   03D |                XX             X   X      X
16 03E | ---------------------------------------------------
17 03F | ---------------------------------------------------
```

2. Mikrotest 1

```
   FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
   Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
       | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
   ----|-----------------------------------------------------|
   BLP:| 88888888888888888888 8888  8888 88888 8888         |
   \   | 99999999999999999999 7777  5555 99999 8888 VCI     |
    \  | 00000000111111322202 0000  0000 22222 1111 BSD     |
SE/TE\ | 234567890123456123910 1234  1234 45678 3421         |
   ---------------------------------------------------------
20 040 |        X   X X   X
   041 |        X
   042 |        X
   043 |        X     X   X
   044 |        X
   045 |        XXXXX X   X
   046 |        XXXXX
   047 |        XXXXX
   048 |        XXXXX
   049 |        XXXXX
   04A |        XXXXX
   04B |        XXXXX
   04C |        XXXXX
   04D |        XXXXX
21 04E |        XXXXX
   04F |        XXXXX
   050 |        XXXXX
   051 |        XXXXX
   052 |        XXXXX
   053 |        XXXXX
   054 |        X
   055 |        X
   056 |           XX
   057 |           XX
   058 |        XXXXX
22 059 |        XXXXX X   X
   05A |             XX
   05B |        XXXXXX
   05C |          XX
   05D |        XXXXX
23 05E |          XX
   05F |        X X   X   X
   060 |        X X   X   X
   061 |          X   X   X
   062 |        XXXXX
   063 |          X
   064 |        X
   065 |          X
   066 |          X
   067 |          X
   068 |          X
   069 |          X
   06A |          XX
   06B |          X
   06C |          X
   06D |          X
   06E |          X
```

2. Mikrotest 1 , Fortsetzung

```
   FU.-| SSCCTIIDDDDDCICWPUCIW UUUU  MMMM FFFFF M MM BBB SM |
   Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
       | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
   ----|-------------------------------------------------|
   BLP:| 888888888888888888888 8888  8888 88888 8888       |
   \   | 999999999999999999999 7777  5555 99999 8888 VCI   |
    \  | 000000001111111322202 0000  0000 22222 1111 BSD   |
SE/TE\ | 234567890123456123910 1234  1234 45678 3421       |
   ----------------------------------------------------
24 06F |       XXXXX
   070 |       XXXXX
   071 |       X
   072 |       X
   073 |         X
   074 |        X  XX
   075 |           X     X
   076 |       XXXXX
   077 |        X   XX   X
   078 |           X
   079 |           X
   07A |           X
   07B |           X
   07C |       XXXXX
   07D |       XXXXX
25 07E |       XXXXX
   07F |       XXXXX
   080 |       XXXXX
   081 |       XXXXX
   082 |       XXXXX
   083 |       XXXXX
   084 |        XXX
   085 |       XXXXX
   086 |        XXX
   087 |        XXX
   088 |     X
   089 |          X
   08A |         X  X
   08B |        XX X
   08C |         X   X   X
   08D |         X   X   X
   08E |         XX  X   X
   08F |         X  X    X
26 090 |         X   X   X
   091 |         X
   092 |           X
   093 |          X      X
   094 |       X     X   X
   095 |       X     X   X
   096 |       X
   097 |       X         X
   098 |            X    X
   099 |            X    X
   09A |            X    X
   09B |             X   X
   09C |             X   X
```

2. Mikrotest 1 , Fortsetzung

```
     FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
     Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
         | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
     ----|-----------------------------------------------------|
     BLP:| 88888888888888888888 8888  8888 88888 8888         |
     \   | 99999999999999999999 7777  5555 99999 8888 VCI     |
      \  | 00000000111111322202 0000  0000 22222 1111 BSD     |
  SE/TE\ | 234567890123456123910 1234  1234 45678 3421         |
```

```
     09D |     X       X    X
     09E |              X   X
     09F |              X   X
     0A0 |              X
  27 0A1 | XX  X   XXX  X   X
     0A2 |         X   X
     0A3 |             X
     0A4 |   X X XX    X    X
     0A5 |     X
  28 0A6 |   X X       X
     0A7 |   X
     0A8 |   X
     0A9 |   X X       X
     0AA |   X X       X
     0AB |   X X
  29 0AC |     X       X
     0AD |     X       X
     0AE |             X
     0AF |     X
     0B0 |   X X       X
     0B1 |     X
     0B2 |     X
  2A 0B3 |     X
     0B4 |   X X
     0B5 |   X X
     0B6 |   X X
     0B7 |   X X
     0B8 |   X
     0B9 |     X
     0BA |   X X
     0BB |   X X
  2B 0BC |    X            X
     0BD |     X
  2C 0BE |     X       X
     0BF |     X       X
     0C0 |     X
     0C1 |     X
     0C2 |     X
     0C3 |     X
     0C4 |     X
     0C5 |     X
     0C6 |     X
     0C7 |     X
     0C8 |     X
     0C9 |     X
     0CA |     X
     0CB |     X
```

2. Mikrotest 1 , Fortsetzung

```
     ---------------------------------------------------------------
     FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
     Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIIC NMMAC A CS UUU BB |
         | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
     ----|---------------------------------------------------|
     BLP:| 88888888888888888888 8888  8888 88888 8888         |
     \   | 99999999999999999999 7777  5555 99999 8888 VCI     |
      \  | 00000000111111322202 0000  0000 22222 1111 BSD     |
  SE/TE\ | 23456789012345612391 0 1234  1234 45678 3421         |
     ---------------------------------------------------------------

     0CC |     X
     0CD |     X
  2D 0CE | XXXX   X    X
     0CF | XXXX   X    X
     0D0 | XXXX   X    X
     0D1 | XXXX   X    X
     0D2 | XXXX        X
     0D3 | XXXX        X
     0D4 | XXXX        X
     0D5 |   X
  2E 0D6 |   X
     0D7 | XXXX        X
     0D8 | XXXX
     0D9 | X
     0DA | X
     0DB |   X
     0DC | XXXX        X
     0DD | XXXX        X
     0DE | XXXX        X
  2F 0DF |   XX
     0E0 |   XX
     0E1 |   XX
     0E2 | X
     0E3 | XXXX        X
     0E4 | XXXX        X
  30 0E5 | XXXX        X
     0E6 | XXXX        X
     0E7 |        X    X
     0E8 |        X    X
     0E9 |        X
     0EA |        X
     0EB |    X   X   X
  31 0EC |        X
     0ED |        X
     0EE |        X
     0EF |        X
  32 0F0 | X   XX  XXX
     0F1 |     X
     0F2 |     X   XX
     0F3 |     X   X
     0F4 |     X
     0F5 |     X
     0F6 |     X
     0F7 | ---------------------------------------------------
     0F8 | ---------------------------------------------------
     0F9 | ---------------------------------------------------
     0FA | ---------------------------------------------------
```

2. Mikrotest 1 , Fortsetzung

```
      ---------------------------------------------------------------
      FU.-| SSCCTIIDDDDDCICWPUCIW UUUU  MMMM FFFFF M MM BBB SM |
      Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
          | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MILDT Y NB SSS AA |
      ----|---------------------------------------------------------|
      BLP:| 8888888888888888888888 8888  8888 88888 8888          |
      \   | 9999999999999999999999 7777  5555 99999 8888 VCI      |
       \  | 0000000011111111322202 0000  0000 22222 1111 BSD      |
   SE/TE\ | 2345678901234561239 10 1234  1234 45678 3421          |
      ---------------------------------------------------------------
      0FB | ---------------------------------------------------------
      0FC | ---------------------------------------------------------
      0FD | ---------------------------------------------------------
      0FE | ---------------------------------------------------------
      0FF | ---------------------------------------------------------
   33 110 |     XXX      X
      111 | X    XX     X
      112 |      XX
      113 |      XX
      114 |      X    X
      115 |      X    X
      116 | ---------------------------------------------------------
      117 | ---------------------------------------------------------
   34 118 |      X
      119 |      X
      11A | ---------------------------------------------------------
      11B | ---------------------------------------------------------
   35 11C |      X
      11D |      XX
      11E |      XX
      11F |       X
      120 |       X
      121 |       X
      122 |       X
      123 |      XX
      124 |      X
      125 | ---------------------------------------------------------
      126 | ---------------------------------------------------------
   36 127 |      XX     X
      128 |       X
      129 |       X
      12A |       X
      12B |       X
      12C |       X
      12D |      XX
      12E |      X
      12F |       X
      130 |       X
   37 131 |      X
      132 |       X   X
      133 |       X   X
      134 |      X    X
      135 |      X     X
      136 |       X    X
      137 |            X
      138 |       X    X
      139 | ---------------------------------------------------------
```

2. Mikrotest 1 , Fortsetzung

```
     ---------------------------------------------------------------
     FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
     Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
         | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
     ----|-------------------------------------------------------|
     BLP:| 888888888888888888888 8888  8888 88888 8888         |
     \   | 999999999999999999999 7777  5555 99999 8888 VCI     |
      \  | 000000001111111322202 0000  0000 22222 1111 BSD     |
  SE/TE\ | 234567890123456123910 1234  1234 45678 3421         |
     ---------------------------------------------------------------
     13A | ---------------------------------------------------------
     13B | ---------------------------------------------------------
     13C | ---------------------------------------------------------
  38 13D |       X    X
     13E |       X    X
     13F |       X    X
     140 |             X
     141 |         XXX X
     142 |         XXX X
     143 | X     X   X X
     144 |       X  XX X
     145 | ---------------------------------------------------------
  39 146 |         XXX X
     147 |       X  X  X
     148 |          XX X
     149 |             X
     14A |             X
     14B |             X
     14C |             X
     14D |       X      X   X
     14E |       X
     14F |       X
     150 |       X
     151 | ---------------------------------------------------------
     152 | ---------------------------------------------------------
  3A 153 |       X
     154 |       X
     155 | X   X       X
     156 | X                X
     157 |             XX
     158 |             XX
     159 |             X
     15A |             X
     15B |             X
     15C |             X
     15D |       X     XX
     15E |             X
     15F |             XX
     160 |             XX
  3B 161 |             XX
     162 | XX          XX    X
     163 |             XX
     164 |             X
     165 |       X     XX
     166 |       X     XX    X
     167 |      XX
  3C 168 |              X
```

2. Mikrotest 1 , Fortsetzung

```
-------------------------------------------------------------
FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
    | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
----|-----------------------------------------------------|
BLP:| 88868888888888888888 8888  8888 88888 8888         |
\   | 99999999999999999999 7777  5555 99999 8888 VCI     |
 \  | 00000000111111322202 0000  0000 22222 1111 BSD     |
SE/TE\ 23456789012345612391 0 1234  1234 45678 3421      |
-------------------------------------------------------------

169 |          X  XX
16A |            XXX
16B |              X  X
16C |             XX    X
16D |                   X
16E |             XX
16F |             XX
```

3. Mikrotest 2

```
   FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
   Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
       | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
   ----|-------------------------------------------------------|
   BLP:| 888888888888888888888 8888  8888 88888 8888         |
   \   | 999999999999999999999 7777  5555 99999 8888 VCI     |
    \  | 000000001111111322202 0000  0000 22222 1111 BSD     |
SE/TE\ | 234567890123456123910 1234  1234 45678 3421         |
   ----------------------------------------------------------
3D 171 |              X
   172 |     X X
   173 |       X   XXX
   174 |          XX X
   175 |     X X
   176 |     X X
   177 |     X X
   178 |-------------------------------------------------------
   179 |-------------------------------------------------------
   17A |-------------------------------------------------------
3E 17B | XX                 X
   17C | XX          XX
   17D |  X                 X
   17E | XX
   17F | XX    X            X                      X     XX
   180 |                                           X
   181 | X
   182 |-------------------------------------------------------
   183 |-------------------------------------------------------
   184 |-------------------------------------------------------
3F 185 | X
   186 |                                          XX
   187 |                                           X
   188 |                                           X
   189 |                                          XX
   18A |                                        XXX
   18B |-------------------------------------------------------
   18C |-------------------------------------------------------
40 18D |  X           X
   18E |  X                                        X
   18F |-------------------------------------------------------
   190 |-------------------------------------------------------
41 191 |  X
   192 |  X
   193 |  X                                       XX
   194 |  X                                       XX
   195 |  X                                       XX
   196 |-------------------------------------------------------
   197 |-------------------------------------------------------
   198 |-------------------------------------------------------
42 199 |  X
   19A |  X                                       XX
   19B |  X                                       XX
   19C |  X                                       XX
43 19D |  X                                       XX
   19E |-------------------------------------------------------
   19F |-------------------------------------------------------
```

3. Mikrotest 2 , Fortsetzung

```
      FU.-| SSCCTIIDDDDDCICWPUCIW UUUU  MMMM FFFFF M MM BBB SM |
      Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
          | LHMMMPCPPFPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
      ----|-----------------------------------------------------|
      BLP:| 88888888888888888888 8888  8888 88888 8888          |
      \   | 99999999999999999999 7777  5555 99999 8888 VCI      |
       \  | 00000000111111322202 0000  0000 22222 1111 BSD      |
   SE/TE\ | 23456789012345612391 0 1234  1234 45678 3421         |
      ----------------------------------------------------------
      1A0 |-----------------------------------------------------
   44 1A1 | XX                                      XX
      1A2 |  X                                      XX
      1A3 |-----------------------------------------------------
      1A4 |-----------------------------------------------------
   45 1A5 |                                         XX
      1A6 |                                         XX
      1A7 |                                          X
      1A8 |  X            X                         XX
      1A9 |     X        X                          XX
      1AA |                                          X
      1AB |                                          X
      1AC |                                          X
   46 1AD |                                          X
      1AE |                                         XX
      1AF |                                          X
      1B0 |                                         XX
      1B1 | XX
      1B2 | XX
   47 1B3 | XX
      1B4 |                                         XX
      1B5 |-----------------------------------------------------
      1B6 |-----------------------------------------------------
   48 1B7 |                                         XX
      1B8 |-----------------------------------------------------
   49 1B9 |                                         XX
      1BA |-----------------------------------------------------
   4A 1BB |                                         XX
      1BC |-----------------------------------------------------
   4B 1BD |                                         XX
      1BE |                                         XX
      1BF |                                         XX
      1C0 |                                         XX
      1C1 |                                         XX
      1C2 |                                         XX
      1C3 |                                         XX
      1C4 |-----------------------------------------------------
      1C5 |-----------------------------------------------------
      1C6 |-----------------------------------------------------
   4C 1C7 |                                         XX
      1C8 |                                         XX
      1C9 |                                         XX
      1CA |                                         XX
      1CB |                                         XX
      1CC |-----------------------------------------------------
      1CD |-----------------------------------------------------
      1CE |-----------------------------------------------------
```

3. Mikrotest 2 , Fortsetzung

```
     ------------------------------------------------------------
     FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
     Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
         | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
     ----|------------------------------------------------------|
     BLP:| 888888888888888888888 8888  8888 88888 8888         |
     \   | 999999999999999999999 7777  5555 99999 8888 VCI     |
      \  | 000000001111111322202 0000  0000 22222 1111 BSD     |
  SE/TE\ | 234567890123456123910 1234  1234 45678 3421         |
     ------------------------------------------------------------
  4D 1CF | X                                          XX
     1D0 | X                                          XX
     1D1 | X                                          XX
     1D2 | X                                          XX
     1D3 |  X                                         XX
     1D4 |  X                                         XX
  4E 1D5 |  X                                         XX
     1D6 | XX           X                             XX
     1D7 | X                                          XX
     1D8 | X                                          XX
  4F 1D9 | X                                          XX
     1DA |------------------------------------------------------
  50 1DB | XX
     1DC | X
     1DD | X
     1DE | X   X
     1DF | X
     1E0 | X
     1E1 | X
     1E2 | X
     1E3 | X
     1E4 | X
     1E5 | X
     1E6 | XX  X       X                              X
     A1E7 |-----------------------------------------------------
     1E8 |------------------------------------------------------
  51 1E9 | X                                          X
     1EA | X                                          X
     1EB | X                                          X
     1EC | X                                          X
     1ED | X                                          X
     1EE | X
     1EF | X
     1F0 | X
     1F1 | X
     1F2 | X   X
     1F3 | X   X
     1F4 |  X
     1F5 |     X
     1F6 |------------------------------------------------------
  52 1F7 |                                            XX
     1F8 |                                            XX
     1F9 |                                          XXXX
     1FA | XX
  53 1FB | XX
     1FC |                                             X
     1FD |                                            XX
```

3. Mikrotest 2 , Fortsetzung

```
     ------------------------------------------------------------
     FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
     Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
         | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
     ----|------------------------------------------------------|
     BLP:| 888888888888888888888 8888  8888 88888 8888        |
     \   | 999999999999999999999 7777  5555 99999 8888 VCI    |
      \  | 000000001111111322202 0000  0000 22222 1111 BSD    |
  SE/TE\ | 234567890123456123910 1234  1234 45678 3421        |
     ------------------------------------------------------------
     1FE |                                          XX
  54 1FF |                                          XX
     200 |                                          XX
     201 |                                          XX
     202 |                                          XX
     203 |   X
     204 |   X
  55 205 |   X
     206 |   X
     207 | X                                                X
     208 |  X           X                                   XX
     209 |-------------------------------------------------------
     20A |-------------------------------------------------------
  56 20B |  X
     20C | X
     20D | X
     20E |  X
     20F |  X
     210 | X                                                XX
     211 |                                                  XX
     212 |                                                  XX
  57 213 |                                                  XX
     214 |  X                                               XX
     215 |  X                                               XX
     216 |  X                                               XX
  58 217 |  X                                               XX
     218 |                                                  XX
  59 219 |                                                  XX
     21A |-------------------------------------------------------
     21B |-------------------------------------------------------
     21C |-------------------------------------------------------
  5A 21D |                                   X X
     21E |                                     X
     21F |                                    X  X
     220 |                                       X
     221 |                                       X
     222 |                                       X
     223 |                                     X
     224 |                                       X
     225 |                                       X
     226 |                                       X
     227 |                                       X
     228 |                                       X
     229 |                                       X
     22A |                                   XXXXX
     22B |                                   XXXXX
     22C |-------------------------------------------------------
```

3. Mikrotest 2 , Fortsetzung

```
     FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
     Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
         | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
     ----|-------------------------------------------------|
     BLP:| 888888888888888888888 8888  8888 88888 8888       |
     \   | 999999999999999999999 7777  5555 99999 8888 VCI   |
      \  | 000000001111111322202 0000  0000 22222 1111 BSD   |
  SE/TE\ | 234567890123456123910 1234  1234 45678 3421       |
     ----------------------------------------------------
  5B 22D |                                   XXX
     22E |                                    X
     22F |                                 X   X
     230 |                                 X
     231 |                                 X
     232 |                                     X
     233 |                                     X
     234 |                                     X
     235 |                                     X
     236 |                                     X
     237 |                                     X
     238 |                                     X
  5C 239 |                                     X
     23A |                                     X
     23B |                                  XX
     23C |                                     X
     23D |                                     X
     23E |                                 X   X
     23F |-------------------------------------------------
     240 |-------------------------------------------------
     241 |-------------------------------------------------
     242 |-------------------------------------------------
     243 |-------------------------------------------------
     244 |-------------------------------------------------
     245 |-------------------------------------------------
     246 |-------------------------------------------------
     247 |-------------------------------------------------
     248 |-------------------------------------------------
  5D 249 |                                     X
     24A |                                     X
     24B |                                     X
     24C |                                     X
     24D |                                     X
     24E |                                 X   X
     24F |                                     X
     250 |                                     X
     251 |                                     X
     252 |                                     X
     253 |                                     X
     254 |                                     X
     255 |                                     X
     256 |                                     X
     257 |                                     X
     258 |                                     X
     259 |                                     X
     25A |                                     X
     25B |                                     X
```

3. Mikrotest 2 , Fortsetzung

```
   FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
   Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NNMAC A CS UUU BB |
       | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
   ----|-------------------------------------------------------|
   BLP:| 888888888888888888888 8888  8888 88888 8888         |
   \   | 999999999999999999999 7777  5555 99999 8888 VCI     |
    \  | 000000001111111322202 0000  0000 22222 1111 BSD     |
SE/TE\ | 234567890123456123910 1234  1234 45678 3421         |
   ----------------------------------------------------------
   25C |                                      X
   25D |                                      X
   25E |                                      X
   25F |                                      X
   260 |                                      X
   261 |                                  X   X
   262 |                                  X   X
5E 263 |                                  X   X
   264 |                                      X
   265 |                                      X
   266 |                                     XX
   267 |                                     X
   268 |                                     X
   269 |                                     X
   26A |                                     X
   26B |                                     X
   26C |                                      X
5F 26D |                                      X
   26E |                                      X
   26F |                                      X
   270 |                                      X
   271 |                                      X
   272 |                                     XX
   273 |                                      X
   274 |                                     XX
   275 |                                     X
   276 |-------------------------------------------------------
60 277 |                                      X
   278 |                                      X
   279 |                                      X
   27A |-------------------------------------------------------
   27B |-------------------------------------------------------
   27C |-------------------------------------------------------
61 27D |                                  X  XX
   27E |
   27F |                                      X
   280 |                                      X
62 281 |                                      X
   282 |                                  X   X
   283 |                                  X   X
   284 |                                  X
   285 |                                  X   X
   286 |                                  X   x
63 287 |                                      X
   288 |                                      X
   289 |                                      X
   28A |                                      X
```

3. Mikrotest 2 , Fortsetzung

```
     ---------------------------------------------------------------
     FU.-| SSCCTIIDDDDDCICWPUCTW UUUU  MMMM FFFFF M MM BBB SM |
     Gr. | BBADBDRBEDCAECLCCSIRC SCMA  SIDC NMMAC A CS UUU BB |
         | LHMMMPCPPPPPHLKSSCBSS IBDI  IRPP MHLDT Y NB SSS AA |
     ----|-------------------------------------------------------|
     BLP:| 8888888888888888888 8888  8088 88888 8888            |
     \   | 9999999999999999999 7777  5555 99999 8888 VCI        |
      \  | 00000001111111322202 0000  0000 22222 1111 BSD       |
  SE/TE\ | 23456789012345612391 0 1234  1234 45678 3421          |
     ---------------------------------------------------------------

     28B |                                XXX
     28C |                                X
     28D |                                X
     28E |                                X
     28F |                                X
     290 |                                X
  64 291 |                                X
     292 |                                X
     293 |                                X
     294 |                                 XX X
     295 |                                 XX X
     296 |                                 XX X
     297 |                                 XX X
     298 |                                 XX X
  65 299 |                                 XX X
     29A |                                 XX X
     29B |                                 XX X
     29C |                                 XX X
     29D |                                 XX X
     29E |                                 XX X
     29F |                                 XX X
     2A0 |                                 XX X
  66 2A1 |                                 XX X
     2A2 |                                 XX X
     2A3 |                                   XX
     2A4 |                                   X
     2A5 |                                   X
     2A6 |-----------------------------------------------------
```

5. PVKAA Grundbefehlstest

5.1. Zusammenfassung

Das Diagnoseprogramm PVKAA (Dokumentationsnr. 1.57.700001.8) ist ein Programm der Ebene 4. Es ist bestimmt für den Test der wichtigsten Befehle bzw. Befehlsmodifikationen sowie grundlegender Interrupt- und Exception-Mechanismen der CPU K 2810 (KAA40) des RVS-Rechnersystems robotron K 1840.

Mit dem Grundbefehlstest (Hardcore Instruction Test) sollen alle Befehle überprüft werden, die der Diagnose-Supervisor enthält, um für alle weiteren Systemtests die korrekte Arbeit des Supervisors mit hoher Wahrscheinlichkeit sichern zu können.

Die möglichen Fehlermitteilungen beinhalten den mnemonischen Befehl, die erwarteten und tatsächlichen Daten bei der Befehlsausführung und den Fehler-PC. Mit dem Programm PVKAA ist keine Fehlerauflösung bis auf Steckeinheiten bzw. noch tiefer möglich!

5.2. Testbedingungen

5.2.1. Hardware

Als minimale Hardware-Konfiguration für den Grundbefehlstest sind notwendig:

- Konsolsubsystem K 2811
- CPU K2810
- Hauptspeicher K 3581.10 (mindestens 64 Kbyte)

5.2.2. Software

Der Grundbefehlstest wird unmittelbar vom Konsolprogramm geladen und gestartet und gestattet keine Parallelarbeit des Supervisors bzw. des Betriebssystems.

5.3. Voraussetzungen

Als fehlerfrei werden die Durchläufe der Tests der Ebene 5 vorausgesetzt, d.h. insbesondere

- die Datenwege zum Laden des Hauptspeichers sind in Ordnung

- 64 Kbyte zusammenhängender Hauptspeicher (beginnend an einer Seitengrenze) sind in Ordnung

- die folgenden Befehle sind in Ordnung: MOVL, MOVZBL, MOVW, XORL3 (Register-Modus), CLRL, TSTL, BISL2, BICL2, BRW, BEQL, HALT, MOVPSL, BICPSW, BISPSW, MFPR, MTPR und INCL

- die folgenden Adressierungsmodi müssen in Ordnung sein: R (=Register-Modus), (PC)+ (=Immediate-Modus) und D(PC) (=Relativ-Modus).

5.4. Bedienungshinweise

Die folgenden Schritte sind zum Laden und Starten des Grundbefehlstests auszuführen (vorausgesetzt wird das geladene Konsolprogramm):

- Einlegen der Diskette mit dem Programm PVKAA.EXE (Grundbefehlstest; Diskette DS5)

- Anhalten der CPU mit

  ```
  >>>H<CR>
  ```

  Es erfolgt die Ausschrift:

  ```
              CPU HALTED
  >>>
  ```

- Initialisierung der CPU mit

  ```
  >>>I<CR>
  ```

  Es erfolgt die Ausschrift:

  ```
              INIT SEQ DONE
  >>>
  ```

- Rücksetzen des Internen Synchronbusses (ISB) mit:

  ```
  >>>U<CR>

  >>>
  ```

- Laden des Grundbefehlstests in den Hauptspeicher mit

  ```
  >>>L PVKAA.EXE<CR>
  ```

  falls das Programm ab Adresse 0 geladen werden soll (loadaddr = 0), bzw. mit

  ```
  >>>L PVKAA.EXE/S:loadaddr<CR>   oder

  >>>L/S:loadaddr PVKAA.EXE<CR>
  ```

  falls das Programm ab Adresse loadaddr geladen werden soll (z.B. im Falle von Speicherfehlern in den ersten 64 Kbyte des Hauptspeichers). Die Adresse loadaddr muß dabei ein gültiger Hexadezimalwert sein und an einer Seitengrenze liegen. Nach fehlerfreiem Laden erfolgt die Ausschrift:

```
LOAD DONE, 0000EE00 BYTES LOADED
>>>
```

Das Laden des Programms PVKAA dauert ca. 60 s

- Nach dem Laden des Grundbefehlstests ist die Adresse loadaddr+FE00 mit dem Kommando

  ```
  >>>D/P loadaddr+FE00 0<CR>
  ```

  zu löschen (keine Arbeit unter AFT /Automatischer Fertigungstest/).

- Starten des Programms mit

  ```
  >>>S 200<CR>   (bzw. loadaddr+200)
  ```

  Das Testprogramm meldet sich mit der Ausschrift:

```
RVS DIAGNOSTIC SOFTWARE

VEB Kombinat Robotron


PVKAA V1.0   Hardcore Instruction Test

Hit any key to continue
```

Für die Weiterarbeit des Programmes ist eine beliebige Taste zu drücken.

Danach läuft der Grundbefehlstest als Dauertest, allerdings nur bis zum Erreichen eines eventuellen Fehlers, ab.

Ein Abbruch dieses Tests und damit eine Rückkehr in das Konsolprogramm erfolgt mit CTRL/P (^P). Danach ist die CPU mit dem Kommando HALT anzuhalten.

Mit den Kommandos CTRL/S bzw. CTRL/Q besteht die generelle Möglichkeit, die Bildschirmausgaben anzuhalten bzw. weiterlaufen zu lassen (entspricht der HOLD SCREEN-Taste).

Alle oben genannten Kommandos sowie der Inhalt der 15 Tests sind im Helpfile PVKAA.HLP (Dokumentationsnr. 1.57.700501.4) enthalten. Zusätzlich sind Informationen über die Arbeit mit Unterbrechungspunkten aufgenommen worden. Dieses File kann mittels

```
>>>@PVKAA.HLP<CR>
```

auf dem Konsolterminal ausgegeben werden.

Während des Programmlaufes ist keine HELP-Funktion möglich!

5.5. Funktionsbeschreibung

5.5.1. Übersicht

Der Grundbefehlstest PVKAA ist ein autonom arbeitendes Programm, das weder unter Steuerung noch parallel zum Supervisor oder zum Betriebssystem laufen kann. Seine Aufgabe ist es, alle diejenigen CPU-Befehle bzw. deren verschiedene Modifikationen zu testen, die im Diagnose-Supervisor enthalten sind. Damit wird die Voraussetzung geschaffen, das Speichertestprogramm PVKAS der Ebene 4 und den Diagnose-Supervisor als Basis für alle weiteren Testprogramme korrekt abzuarbeiten. Zusätzlich werden eine Reihe von Exception- und Interruptmechanismen der CPU überprüft. Der Grundbefehlstest ist als Dauertest angelegt und gestattet nur wenige Möglichkeiten der Bedienerkommunikation.

5.5.2. Programmgröße

Der Grundbefehlstest benötigt für seine Abarbeitung insgesamt 64 Kbyte zusammenhängenden Hauptspeicher, wobei für das eigentliche Programm 59 Kbyte reserviert sind. Nachfolgend wird eine kurze Übersicht über die Speicheraufteilung (jeweils bezogen auf die Ladeadresse) gegeben (Adressangaben hexadezimal):

Ab Adresse	Inhalt
0200	Definitionen
033D	Test-Daten
17E2	Definitionen (u.a. System Control Block)
2000	Trapbehandlungsroutinen
2491	Tabellen für Schleifenadressen usw.
287D	Fehlerausschriften und Mitteilungen
2A9A	Segmente für Fehlerausschriften
4102	System-/Programm-Initialisierung
4627	Routinen zur Unterbrechungspunktbehandlung sowie Tabellen und Ausschriften
4C1A	Tests 1 bis 15
B8D3	Allgemeine Unterprogrammme
COC6 (- ECC5)	Daten
ECC6 (- EDFF)	Routinen zur Unterbrechungspunktbehandlung
FE00 (- FFFF)	AFT-Steuerblock.

5.5.3. Programmlaufzeit und Programmitteilungen

Die Programmlaufzeit für einen Durchlauf der Tests 1 bis 15 beträgt ca. 0.3 s.

Am Ende des ersten und anschließend nach jeweils 44 (=%X2C) Durchläufen des Grundbefehlstests (d.h. nach jeweils etwa 12 s) erfolgt die Programmausschrift:

```
PVKAA 1.0 pass xxxx(X) done!
```

Diese Ausschrift kann bei Langzeittests durch Eingabe von ^X (CTRL/X) aus- bzw. wiedereingeschaltet werden. Je nach Programmzustand folgt dann eine der beiden Meldungen:

```
^X
EOP's are now disabled - PVKAA is STILL running
```

oder

```
^X
EOF's are reenabled
```

5.5.4. Abarbeitungsbedingungen

Wie bereits in 5.4. erläutert, ist es möglich, den Test an eine beliebige Speicherposition zu laden, vorausgesetzt, daß dieser Speicherbereich 64 Kbyte umfaßt, zusammenhängend ist und an einer Seitengrenze beginnt.

5.5.5. Flag-Verwendung

Entfällt.

5.5.6. Allgemeine Programmanforderungen

Entfällt.

5.5.7. Fehlermitteilungen

Im Falle von Fehlern wird vom Grundbefehlstest die folgende allgemeine Fehlermitteilung gegeben:

```
??? ERROR - Test # xx, Subtest # xx[, Subsubtest # xx] xxxxx failed
xxx...xxx
Expected - xxxxxxxx
Received - xxxxxxxx
        HALT INST EXECUTED
        HALTED AT xxxxxxxx

>>>
```

bzw. wenn keine speziellen Datenvergleiche vorliegen:

```
No expected/received data
```

Die erste Zeile enthält die Test-, Subtest- und gegebenenfalls die Subsubtestnummer sowie den mnemonischen Befehlscode, bei dem dieser Fehler auftrat.
Die zweite Zeile beinhaltet eine ausführliche Fehlerinformation über ermittelte bzw. mögliche Fehlerursachen. Insgesamt existieren über 60 derartige Mitteilungen, z.B.

```
Condition Code bits wrong      oder

Source 2 is incorrect
```

Die dritte und vierte Zeile enthalten, wenn vorhanden, die erwarteten sowie die tatsächlich aufgetretenen Daten. Die letzte (sechste) Zeile enthält den Wert des Programmzählers (PC), an dem das Programm beendet wurde. Dabei wird die Adresse angezeigt, die dem HALT-Befehl folgt. Im Falle des normalen Ladens ab Adresse 0 entspricht dieser PC-Wert exakt den Adressangaben in der Programmliste, anderenfalls ist die jeweilige Verschiebung (=loadaddr) zu subtrahieren.

Nach Übergabe der Steuerung an das Konsolprogramm besteht die Möglichkeit, an der angezeigten Fehlerposition eine Fehlerschleife mittels der beiden Konsolkommandos

>>>D/P/B xxxxxxxx 1<CR>

>>>C<CR>

zu generieren. xxxxxxxx ist dabei der in der letzten Zeile der Fehlermitteilung ausgegebene und um 1 verringerte Fehler-PC. An der betreffenden Speicherposition wird dabei durch das DEPOSIT-Byte-Kommando der HALT-Befehl durch NOP (Operationscode 01) ersetzt und damit eine geeignete Schleife aufgebaut und die Verarbeitung mit CONTINUE (C) fortgesetzt. Alle weiteren Fehlermeldungen werden danach unterdrückt, um eine für eventuell folgende Messungen zeitoptimale Schleife zu erhalten.

Auf Fehlermitteilungen im Zusammenhang mit der Arbeit mit Unterbrechungspunkten wird im folgenden Abschnitt eingegangen.

5.5.8. Arbeit mit Unterbrechungspunkten

Auf Grund der Stand-Alone-Arbeitsweise des Diagnoseprogramms PVKAA müssen sämtliche Ein- und Ausgaben von und auf die BDE K8911.80 vom Programm selbst über die Prozessorregister RXCS (=%X20), RXDB (=%X21), TXCS (=%X22) und TXDB (=%X23) organisiert und ausgeführt werden. Insbesondere sind dabei verschiedene Ausgabekonvertierungen, die Analyse von Tastatureingaben sowie eine Reihe von Steuerungsaspekten zu berücksichtigen.
Aus diesen Aufwandsgründen sind die Möglichkeiten der Bedienerkommunikation gegenüber anderen Diagnoseprogrammen, z.B. den unter dem Diagnose-Supervisor arbeitenden Programmen (bei denen der Supervisor alle Ein- und Ausgaben behandelt), stark eingeschränkt.

Außer den o.g. Fehlerschleifen wird als einzige Variante der Beeinflussung des Ablaufs des Testprogramms PVKAA die Arbeit mit Unterbrechnungspunkten unterstützt.
Zusammen mit den Kommandos, die das Konsolprogramm bietet, besteht dennoch die Möglichkeit, sich effektiv an etwaige Fehler heranzutasten.

5.5.8.1. Setzen und Löschen von Unterbrechungspunkten

Bei der Arbeit mit dem Programm PVKAA bestehen zwei Varianten, das Setzen bzw. Löschen von Unterbrechungspunkten zu erreichen:

- das Programm wird mittels

  >>>S 220<CR> (bzw. loadaddr+220)

  gestartet oder

- durch Eingabe von CTRL/C (^C) an einem beliebigen Punkt der Abarbeitung.

Damit wird durch das Programm PVKAA ein Unterprogrammsprung in die Routine BPT_SET_BR_PNT realisiert, die das Setzen und Löschen von Unterbrechungspunkten behandelt. Es erfolgt die Kommandoanforderung

Input breakpoint:

Danach bestehen für den Anwender die folgenden Möglichkeiten der Eingabe (xxxx muß dabei jeweils eine gültige Hexadezimalzahl sein; dabei werden die Ziffern "a" bis "f" nicht akzeptiert, sondern nur "A" bis "F"):

- xxxx<CR> Setzen eines Unterbrechungspunktes auf die Adresse xxxx. Bei loadaddr= 0 ist die Adresse ohne Verschiebung (loadaddr) einzugeben!

- Xxxxx<CR> Löschen des Unterbrechungspunktes auf der Adresse xxxx. Bei loadaddr= 0 ist die Adresse ohne Verschiebung (loadaddr) einzugeben!

  Bem.: Bei Eingabe von mehr als 4 Ziffern werden nur die jeweils letzten 4 Ziffern berücksichtigt; so ergibt z.B. die Eingabezeichenfolge "458A26" den Adresswert 8A26.

- X<CR> Löschen aller Unterbrechungspunkte.

- CTRL/R (^R) Ignorieren der gesamten Eingabezeile; es erfolgt eine erneute Kommandoanforderung.

- <DEL> Durch (eventuell mehrfache) Eingabe von <DEL> wird das bzw. die letzte(n) eingegebene(n) Zeichen gelöscht; für jede gelöschte Ziffer wird ein Backslash "\" ausgegeben, jedoch nicht die gelöschte Ziffer selbst!
  Werden nur die Ziffern gelöscht, so erfolgt keine neue Anforderung; wird zusätzlich auch "X" gelöscht (ohne Anzeige von "\"!), so wird eine erneute Eingabeanforderung auf dem Konsolterminal ausgegeben.

- <CR> Mit der Eingabe von <CR> wird diese Behandlungsroutine beendet und in das laufende Testprogramm zurückgekehrt.

- Sonstiges Bei allen anderen Zeicheneingaben außer den o.g. erfolgt die Fehlerausschrift

  ???ERROR - illegal character

  und eine erneute Kommandoanforderung.

Maximal können 100 Unterbrechungspunkte gesetzt werden; dabei ist zu beachten, daß sowohl beim Setzen als auch beim Löschen kein Test auf einen Überlauf der Tabelle der Unterbrechungspunkte bzw.auf das Ende dieser Tabelle gemacht wird!

5.5.8.2. Erreichen eines Unterbrechungspunktes

Wird durch das Testprogramm ein Unterbrechungspunkt erreicht, so erfolgt auf dem Bildschirm die Mitteilung

Breakpoint at xxxx

wobei xxxx die erreichte Adresse darstellt.

Danach stehen folgende Möglichkeiten der Weiterarbeit dem Nutzer zur Auswahl:

- CTRL/C (^C) Bei Eingabe von CTRL/C geht das Testprogramm PVKAA mit der Ausschrift

```
^C
        HALT INST EXECUTED
        HALTED AT 00004B7D   (bzw. loadaddr+4B7D)

>>>
```

in den Halt-Zustand und die Steuerung wird an das Konsolprogramm übergeben. Danach können alle Konsolkommandos wie EXAMINE, DEPOSIT usw. verwendet werden. Besonders sei hier auf

SET STEP INSTRUCTION (SE S I) und
CONTINUE (C)

sowie

CLEAR STEP (CL S)

verwiesen, mit denen der Schrittbetrieb auf Befehlsniveau im Testprogramm realisiert werden kann bzw. wieder aufgehoben wird.

- CTRL/R (^R) Mit der Eingabe von CTRL/R wird die im vorangegangenen Abschnitt beschriebene Routine zum Setzen und Löschen von Unterbrechungspunkten aktiviert:

```
^R
Input breakpoint:
```

- Sonstiges Bei allen anderen Zeicheneingaben außer den beiden genannten wird das Programm fortgesetzt; der Unterbrechungspunkt bleibt dabei erhalten.

5.5.8.3. Ausgabe der Tabelle der Unterbrechungspunkte

Befindet sich das Testprogramm PVKAA im Halt-Zustand (z.B. nach Erreichen eines Unterbrechungspunktes und Eingabe von CTRL/C), so kann durch das Starten eines autonomen Zusatzprogramms auf der Adresse 214 die Tabelle der Unterbrechungspunkte und der jeweiligen geretteten Operationscodes auf dem Konsolterminal ausgegeben werden. Am Ende der Tabelle wird - falls Unterbrechungspunkte enthalten sind - die Basisadresse loadaddr angezeigt. Danach geht die Steuerung wieder an das Konsolprogramm zurück.

```
>>>S 214<CR>   (bzw. loadaddr+214)

        Breakpoint Table PVKAA V1.0

        xxxx xx
        xxxx xx
        xxxx xx

        BAS= xxxxxxxx

        HALT INST EXECUTED
        HALTED AT 0000ED0F   (bzw. loadaddr+ED0F)
```

>>>

Bem.: Eine zusätzliche Ausgabe von (*) kennzeichnet mehrfach eingetragene Unterbrechungspunkte, da bei der Eingabe keine Kontrolle auf bereits existierende gleiche Unterbrechungspunkte erfolgt. Diese mehrfach eingetragenen Unterbrechungspunkte werden bei Abarbeitung dieses Zusatzprogrammes automatisch in der Tabelle gelöscht!

5.6. Testbeschreibungen

5.6.1. Test 1 - Test grundlegender Befehle

Dieser Test umfaßt 20 Subtests und 14 Subsubtests (im Subtest 8). In diesem ersten Test werden zunächst mit Hilfe der Befehle BICPSW, BISPSW und MOVPSL die Bedingungscode-Bits geprüft (Subtest 1), anschließend werden die Befehle JSB, BSBW und BSBB (Subtests 2 bis 4) getestet.
Die Subtests 5 bis 7 enthalten grundlegende Tests der Befehle CMPB, CMPW und CMPL.
Die 14 Subsubtests des Subtests 8 prüfen nacheinander die Register R0 bis R13 auf ihre Beschreibbarkeit.
Der AOBLEQ-Befehl wird im Subtest 9 getestet, danach im Subtest 10 BITL mit unterschiedlichen Adressierungsmodi. ROTL wird im nachfolgenden Subtest 11 ebenfalls in mehreren Adressierungsarten überprüft; dabei werden die Quelloperanden, Zieloperanden sowie die zu erwartenden Daten, Bedingungsbits usw. aus im Programm definierten Tabellen (Test-Daten, s. 5.5.2.) entnommen bzw. mit diesen verglichen.
Die Subtests 12 bis 14 enthalten die Tests der Befehle MOVZBW, MOVZBL und MOVZWL. Die Befehle TSTB, TSTW, TSTL, CLRL, MOVB und MOVW werden in den Subtests 15 bis 20 jeweils für mehrere Adressierungsmodi geprüft.

5.6.2. Test 2 - Tests bedingter und sonstiger Verzweigungsbefehle

Der Test 2 umfaßt 10 Subtests und 13 Subsubtests (Subtest 1). Der erste Subtest enthält nacheinander in den einzelnen Subsubtests eine Überprüfung aller bedingten (und der unbedingten) Verzweigungsbefehle BRB, JMP, BLEQ, BLSS, BLEQU, BVS, BCS, BGTR, BNEQ, BGEQ, BGTRU, BVC und BCC.
Die anderen 9 Subtests testen jeweils einen der nachfolgenden Verzweigungsbefehle BBC, BBS, BBSS, BBCS, BBSC, BBCC, BBCCI, BLBS und BLBC. Die Subtests 2 bis 8 testen dabei jeweils alle 32 möglichen Bitpositionen.

5.6.3. Test 3 - Tests von Befehlen mit variabler Bitfeldverarbeitung

In den 5 Subtests werden die Befehle INSV, EXTV, EXTZV, CMPV und CMPZV überprüft.

5.6.4. Test 4 - Adressierungstest

In den 5 Subtests erfolgt an Hand von Datentabellen der Test der Adressierungsarten Autoinkrement, Autodekrement, Index, Registerschreiben und Speicherschreiben.

5.6.5. Test 5 - Operandenspezifikationstest

In den insgesamt 63 Subtests dieses Tests werden eine Reihe von arithmetischen, logischen und Transportbefehlen mit den in Test 4 überprüften Adressierungsmodi getestet.

5.6.6. Test 6 - Test der Befehle AOBLSS und ACBL

In den 2 Subtests erfolgt die Prüfung der Befehle AOBLSS und ACBL.

5.6.7. Test 7 - Test der Befehle SOBGEQ und SOBGTR

Die 2 Subtests nehmen Überprüfungen der beiden Befehle SOBGEQ und SOBGTR vor.

5.6.8. Test 8 - Test der Befehle CASEB, CASEW und CASEL

Die Befehle CASEB, CASEW und CASEL werden in den 3 Subtests des Tests 8 auf grundlegende Richtigkeit überprüft.

5.6.9. Test 9 - Test der Integerarithmetik- und Logik-Befehle

Dieser Test umfaßt insgesamt 70 Subtests, mit denen die Integerarithmetik- und logischen Befehle jeweils für die verschiedenen Datentypen B, W und L als auch in ihren 2- bzw. 3-Operanden-Versionen getestet werden. Für die Befehle ADD, SUB, MUL, DIV, XOR, BIC und BIS ergeben sich damit jeweils 6 Subtests. Desweiteren werden Überprüfungen der Befehle ADWC, ASHx, BITx, CLRQ, DECx, INCx, EDIV, EMUL, MNEGx, MCOMx, MOVQ, MOVCx, PUSHL, SBWC, CMPC5 sowie CVTBL, CVTWL und CVTBW vorgenommen.

5.6.10. Test 10 - Prozeduraufruftests

Im Subtest 1 werden der Prozeduraufruf und die Rückkehr aus dieser Prozedur mittels Argumentlistentechnik (CALLG) getestet, der zweite Subtest überprüft den Prozeduraufruf mittels Parameterübermittlung im Stack (CALLS).

5.6.11. Test 11 - Tests sonstiger Befehle

Die jeweils 2 Subsubtests der Subtests 1 und 2 nehmen eine Überprüfung der Befehle PUSHR und POPR mit bzw. ohne Test des SP-Registers vor.
FFC und FFS werden im Subtest 3 und 4 getestet.
Die folgenden 7 Subtests 5 bis 11 testen den PUSHAx-Befehl in seinen möglichen Variationen x=B, D, F, G, L, Q und W.
Der Subtest 12 führt einen Test des Befehls LOCC durch.
Danach folgt der Test von MOVAx (x=B, D, F, G, Q, W; Subtests 13 bis 18).

Abschließend wird ein Test des MOVZBW-Befehls (Subtest 19) vorgenommen.

5.6.12. Test 12 - Exception- und Interrupttests

In den ersten 3 Subtests erfolgt der Test von Exceptions; die Subtests 4 bis 19 beinhalten die Tests von Interrupthehandlungen.

5.6.13. Test 13 - Taktintervalltest

Dieser Test umfaßt insgesamt 5 Subtests, die folgende einzelne Teiltests vornehmen: NICR- (Next Interval Count Register) und ICR- (Interval Count Register) Test, Carry-Logiktest, Fehlerbittest, Freilauftest und Interrupttest.

5.6.14. Test 14 - Test der Warteschlangenbefehle

4 der insgesamt 8 Subtests überprüfen den INSQUE-Befehl, die restlichen vier den REMQUE-Befehl.

5.6.15. Test 15 - Konsol-Transmitter-Test

Die 3 Subtests des abschließenden Tests nehmen eine Überprüfung des DONE-Bits und der Interruptlogik zur Konsolausgabe vor.

5.7. Auflistung der Tests

Test 1 - Test grundlegender Befehle
Test 2 - Tests bedingter und sonstiger Verzweigungsbefehle
Test 3 - Tests von Befehlen mit variabler Bitfeldverarbeitung
Test 4 - Adressierungstest
Test 5 - Operandenspezifikationstest
Test 6 - Test der Befehle AOBLSS und ACBL
Test 7 - Test der Befehle SOBGEQ und SOBGTR
Test 8 - Test der Befehle CASEB, CASEW und CASEL
Test 9 - Test der Integerarithmetik- und Logik-Befehle
Test 10 - Prozeduraufruftests
Test 11 - Tests sonstiger Befehle
Test 12 - Exception- und Interrupttests
Test 13 - Taktintervalltest
Test 14 - Test der Warteschlangenbefehle
Test 15 - Konsol-Transmitter-Test

6. Grundspeichertest

6.1. Zusammenfassung

Das Diagnoseprogramm PVKAS (Dokumentationsnummer 1.57.700002.8) ist ein Programm der Ebene 4.
Es ist für den Test des Hauptspeichers K 3581 der Rechnersysteme K 1840 und K 1845 bestimmt.
Mit dem Diagnoseprogramm PVKAS wird die Funktionsfähigkeit des gesamten Hauptspeichers geprüft. Als Fehlermitteilungen werden jeweils

- der aktuell abzuarbeitende Test sowie Subtest,
- der Fehlertyp,
- die Speicheradresse, an welcher der Fehler auftrat,
- der Inhalt der Speicherkonfigurationsregister zum Zeitpunkt des Auftretens eines Speicherfehlers sowie
- der bzw. die fehlerhaften Schaltkreise

ausgegeben.
Dieses Diagnoseprogramm läuft im stand-alone Modus ohne Zuhilfenahme des Diagnose-Supervisors und gestattet keine Parallelarbeit des Betriebssystems.

6.2. Testbedingungen

6.2.1. Hardware

Zur Abarbeitung des Diagnoseprogrammes PVKAS ist mindestens folgende Hardwarekonfiguration erforderlich:

- Konsolsubsystem	KSS K 2811
- Zentrale Verarbeitungseinheit	CPU K 2810
- Hauptspeicher	MEM K 3581.10 (RVS K 1840) bzw.
	MEM K 3581.20 (RVS K 1845)
(Speichergrundkonfiguration)	

6.2.2. Software

Das Diagnoseprogramm PVKAS wird vom Konsolsubsystem aus in den Hauptspeicher geladen und gestartet.
Damit die Ausführung aller möglichen Speicherfunktionen erzwungen werden kann, wird während des Programmlaufes der Zugriff zum Cache verhindert.

6.3. Voraussetzungen

- Fehlerfreier Durchlauf der Konsoltests, der Mikrodiagnosetests und des Grundbefehlstests.
- 17.5 Kbyte bzw. 35 Kbyte (s. Abschn. 6.5.2.) zusammenhängender Hauptspeicher.

6.4. Bedienungshinweise

Unter der Voraussetzung, daß das Konsolprogramm bereits geladen ist, sind zum Laden und Starten des Diagnoseprogrammes PVKAS folgende Schritte auszuführen:

- Einlegen der Diskette mit dem Programm PVKAS.EXE (Diskette DS5).

- Anhalten der CPU mit:

  >>>H

  Es erscheint die Ausschrift:

  CPU HALTED
  >>>

- Initialisierung der CPU mit:

  >>>I

  Es erscheint die Ausschrift:

  INIT SEQ DONE
  >>>

- Rücksetzen des Internen Synchronbusses (ISB) mit:

  >>>U
  >>>

- Laden des Diagnoseprogrammes in den Hauptspeicher mit:

  >>>L PVKAS.EXE

  falls das Programm ab der Adresse 0 geladen werden soll (loadaddr = 0) bzw. mit:

  >>>L PVKAS.EXE/S:loadaddr oder
  >>>L/S:loadaddr PVKAS.EXE

  falls ab der Adresse loadaddr geladen werden soll.
  Loadaddr muß ein gültiger Hexadezimalwert sein und eine Seitengrenze kennzeichnen. Nach fehlerfreiem Laden erscheint die Ausschrift:

  LOAD DONE, 0000XX00 BYTES LOADED
  >>>

- Starten des Programmes:

Das Programm kann ab der Startadresse loadaddr+200 oder der Startadresse loadaddr+204 abgearbeitet werden. Während im ersten Fall die sofortige Testdurchführung gefordert wird, kann im zweitem Fall der Bediener die Testbedingungen vor dem eigentlichen Test verändern.

Fall 1: Starten mittels Konsolkommando

```
>>>S 200          (bzw. loadaddr+200)
```

Beim erstmaligem Start nach einem Ladevorgang mittels Kommando >>>S 200 nimmt das Programm an, daß der gesamte vorhandene Hauptspeicher getestet werden soll. Bei allen nachfolgenden Starts (auch mittels >>>S 204) ist der zu testende Speicherbereich durch den aktuellen Inhalt der programmspezifischen Steuerregister CMDREG 0 und CMDREG 1 (s. Abschn. 6.5.9.) festgelegt.

Das Programm meldet sich mit der Ausschrift:

```
                    RVS DIAGNOSTIC SOFTWARE

                     VEB KOMBINAT ROBOTRON

PVKAS VX.X   Memory Test  (xx/xx/xx)

started at xx:xx:xx.xx

Memory at TR-level      : xx
Memory Chips            : xxK RAM's (x MB arrays)
Memory mode             : xxxxxxxxx
Memory system capacity  : xx Mbyte
```

Dabei gibt (xx/xx/xx) das Erstellungsdatum der genutzten Programmversion an. Anschliessend wird mitgeteilt, ob mit oder ohne Umlagerung des Programmes während eines Testlaufes gearbeitet wird. Ergänzt wird diese Mitteilung um die Angabe des aktuellen Speicherbereiches, in dem sich das Diagnoseprogramm befindet sowie dem aktuellen Testbereich in folgender Form:

```
Actual test area         : xxxxxxxx ... xxxxxxxx
Actual programm area     : xxxxxxxx ... xxxxxxxx
```

Fall 2: Starten mittels Konsolkommando

```
>>>S 204          (bzw. loadaddr+204)
```

Das Programm meldet sich mit der Ausschrift:

```
CMDREG 0  MEMB   = xxxxxxxx
CMDREG 1  MEME   = xxxxxxxx
CMDREG 2  TEST   = xxxxxxxx
CMDREG 3  PERM   = xxxxxxxx
CMDREG 4  ERRHD  = xxxxxxxx
CMDREG 5  ERRPN  = xxxxxxxx
CMDREG 6  ERRLP  = xxxxxxxx
INPUT CMDREG (0|...|6|<CR>|^C) :
```

und erwartet entsprechende Bedienereingaben (s. Abschn. 6.5.9.).
Mittels der Bedienereingaben können die Testabarbeitungsbedingungen variiert werden, indem die programmspezifischen Steuerregister CMDREG x (command register x) mit definierten Bitkombinationen beschrieben werden (s. Abschn.

6.5.9.).
Nach Abschluß der Bedienereingaben meldet sich das Programm mit den unter "Starten des Programmes mit:>>>S 200" angebenen Ausschriften und der Angabe des aktuell genutzten Arbeitsmodus (s. Abschn. 6.5.1.).
Daran anschließend wird der Grundspeichertest in Abhängigkeit vom Inhalt der programmspezifischen Steuerregister CMDREG x ausgeführt.

- Testabbruch:

  Ein Testabbruch und damit die Rückkehr in das Konsolprogramm wird durch Eingabe von CTRL/P erzwungen. Anschließend ist die CPU mit dem Kommando HALT anzuhalten. Werden durch die nachfolgenden Bedienerhandlungen keine Informationen verfälscht, die für die korrekte Arbeit des Diagnoseprogrammes erforderlich sind (Speicherkonfigurationsregister, Programmvariable), ist durch Eingabe des Konsolkommandos:

  >>>C

  die Fortführung des Grundspeichertests möglich. Damit kann zugleich die Fehleranalyse und -ortung vervollständigt bzw. bestätigt werden.

Mit den Kommandos CTRL/S bzw. CTRL/Q besteht generell die Möglichkeit, die Bildschirmausgaben anzuhalten bzw. weiterlaufen zu lassen (entspricht der HOLD SCREEN Taste).
Alle oben genannten Kommandos sowie die Beschreibung der programmspezifischen Steuerregister sind im Helpfile PVKAS.HLP enthalten. Dieses File kann mittels

>>>@PVKAS.HLP

auf dem Konsolterminal ausgegeben werden. Während des Programmlaufes ist keine Helpfunktion möglich.

6.5. Funktionsbeschreibung

6.5.1. Übersicht

Das Diagnoseprogramm PVKAS ist ein autonom arbeitendes Programm, das weder unter Steuerung noch parallel zum Diagnose-Supervisor (DS) oder zum Betriebssystem läuft. Seine Aufgabe ist es, die Funktionsfähigkeit des Hauptspeichers K3581 nachzuweisen bzw. Fehler bei der Ausführung von Speicherfunktionen (z.B. einfaches/ erweitertes Schreiben und Lesen, Refreshfunktion) zu erkennen und mitzuteilen und möglichst genau den Fehler zu lokalisieren (Speicher- bzw. Korrekturschaltkreise).
Die fehlerfreie Abarbeitung des Grundspeichertests bildet somit sowohl die Grundlage für die korrekte Arbeit des DS bzw. des Betriebssystems als auch aller Diagnoseprogramme der Ebenen 3, 2 und 1.
Der Grundspeichertest kann als Dauertest oder als Einzeltest abgearbeitet werden. Da das Diagnoseprogramm PVKAS keine Dienste des DS in Anspruch nimmt, ist eine spezielle Bedienerkommunikation (s. Abschn. 6.5.9.) vorgeschrieben. Mit Hilfe dieser Bedienerkommunikation können

- der konkrete Testablauf (z.B. Dauertest oder Einzeltest, Halt nach Fehler bzw. kein Halt nach Fehler),
- der Umfang der Programm- und der Fehlermitteilungen sowie
- das Testobjekt (Bereichsgrenzen des Prüfbereiches im Hauptspeicher)

festgelegt werden.

Das Diagnoseprogramm kennt zwei unterschiedliche Arbeitsmodi:

- den Umlagerungsmodus und
- den Modus ohne Umlagerung.

Der Arbeitsmodus wird vom Diagnoseprogramm selbsttätig eingestellt und ist vom Bediener nur indirekt beeinflußbar:
Unter der Bedingung, daß das Diagnoseprogramm ab der Adresse 0 geladen wurde, beim erstmaligen Starten des Programmes das Konsolkommando >>>S 200 genutzt wurde und der gesamte vorhandene Hauptspeicher geprüft werden soll, wird der Umlagerungsmodus eingestellt. Hierbei lagert sich das Programm selbsttätig auf definierte Speicherbereiche um und gestattet somit das Testen des gesamten Speichers. In diesem Modus kann der Bediener die Testbereichsgrenzen (s. Abschn. 6.5.9.) dynamisch, d.h. während des Testlaufes, nicht ändern.

Ist eine der oben angegebenen Bedingungen verletzt, wird der Modus ohne Umlagerung gewählt. Hierbei prüft das Programm, daß sich der Testbereich und der Programmbereich nicht überschneiden. In diesem Modus kann der Bediener alle Steuerregisterinhalte (CMDREG x, s. Abschn 6.5.9.) dynamisch ändern.

6.5.2. Programmgröße

Das Diagnoseprogramm PVKAS hat eine Größe von 17.5 Kbyte. Für seine Abarbeitung werden im Modus ohne Umlagerung 17.5 Kbyte und im Umlagerunsmodus 35 Kbyte zusammenhängender Speicherplatz benötigt.

6.5.3. Programmlaufzeit

Die Programmlaufzeit beträgt für einen fehlerfreien Durchlauf der Tests 1, 2 und 3 im Modus ohne Umlagerung 0:45 + X * 1:50 min. Die Variable X steht für die Anzahl der zu testenden Mbyte Hauptspeicherbereich. Wird im Umlagerungsmodus gearbeitet, erhöht sich der erhaltene Wert um 0:45 min.
In Abhängigkeit von der Fehlerzahl und dem gewünschten Umfang an Programm- und Fehlermitteilungen (s. Abschn 6.5.5.) kann sich die Programmlaufzeit aufgrund der Fehleranalyse (auch bei ausgeschalteter Fehlerausgabe) auf ein Vielfaches des oben genannten Richtwertes erhöhen.
Für einen fehlerfreien Durchlauf des Tests 4 werden minimal 10 min benötigt.

6.5.4. Abarbeitungsbedingungen

Das Diagnoseprogramm PVKAS kann ab beliebigen, Seitengrenzen kennzeichnenden Speicheradressen geladen werden, vorausgesetzt, daß 17.5 Kbyte (bzw. 35 Kbyte) zusammenhängender Speicherplatz verfügbar sind.
Um die Funktionsfähigkeit des Hauptspeichers überprüfen zu können, wird während des Programmlaufes

- der Zugriff zum Cache verhindert und
- die selbsttätige Korrektur von 1-Bit-Fehlern durch den Hauptspeicher zugelassen.
  Beachte: Das Diagnoseprogramm muß in den Hauptspeicher geladen und von dort aus abgearbeitet werden. Bei Verbot der Korrektur von 1-Bit-Fehlern kann eine Datenverfälschung im Programmbereich zu undefinierten Zuständen des Diagnoseprogrammes bzw. des Programmablaufes führen.

6.5.5. Fehlerermittlung

Tritt während des Grundspeichertests ein Fehler auf, so wird entsprechend dem gewünschten Umfang an Fehlerinformationen jeweils eine Fehlermitteilung ausgegeben.
Die Fehlermitteilungen können unterdrückt werden (günstig zum schnellen Erreichen bestimmter Speicherzellen, s. Abschn. 6.5.9.).
Auf der Grundlage der Fehlermitteilungen sowie einer Analyse des Inhaltes der Speicherkonfigurationsregister (memory configuration register x - MCRx) kann ein Fehler lokalisiert werden. Dieser Prozeß wird effektiv unterstützt, wenn die Tests 1 bis 3 mindestens bis zum Vorliegen eines Fehlers abgearbeitet werden (aufgrund der unterschiedlichen Prüfmuster in den einzelnen Tests sind jeweils nur bestimmte Fehlerquellen lokalisierbar).

Das Format einer gekürzten Fehlermitteilung hat folgendes Aussehen:

```
??? ERROR - Test #xx, Subtest #xx
Fehlertyp
Address          : xxxxxxxx
```

Bei Forderung nach vollständiger Fehlerausgabe erscheinen darüberhinaus die Zeilen

```
Expected data    : xxxxxxxx
Received data    : xxxxxxxx
XOR              : xxxxxxxx
MCRA [xxxxxxxx]  : xxxxxxxx
MCRB [xxxxxxxx]  : xxxxxxxx
MCRC [xxxxxxxx]  : xxxxxxxx
MCRD [xxxxxxxx]  : xxxxxxxx
MCRE [xxxxxxxx]  : xxxxxxxx
MCRF [xxxxxxxx]  : xxxxxxxx
```

Wird beim Auftreten eines Fehlers der Übergang des Programmes in den Haltzustand gefordert (s. Abschn. 6.5.9.), so folgt die Mitteilung:

```
HALT INST EXECUTED
HALTED AT xxxxxxxx
```

Die erste Zeile einer Speicherfehlermitteilung gibt die Nummer des Tests und des Subtests an, in welchem der Fehler auftrat. Die zweite und dritte Zeile spezifizieren den Fehlertyp und die fehlerhafte Speicheradresse. Auf die jeweils zu prüfende Speicherzelle bezogen, werden vom Diagnoseprogramm PVKAS folgende speicherspezifischen Fehlertypen unterschieden:

SINGLE BIT ERROR : vom Hauptspeicher wurde ein Ein-Bit-Fehler erkannt und korrigiert

MULTIPLE-BIT-ERROR : vom Hauptspeicher wurde ein Mehr-Bit-Fehler erkannt

SING & MULT BIT ERROR : vom Hauptspeicher wurden sowohl ein Ein-Bit-Fehler als auch ein Mehr-bit-Fehler erkannt (d.h. auf verschiedenen Speicherbanks liegen zugleich Ein-Bit- bzw. Mehr-Bit-Fehler vor)

DOUBLE BIT ERROR : es liegt ein Mehr-Bit-Fehler vor, der waehrend der Fehleranalyse als ein Zwei-Bit-Fehler identifiziert werden konnte

COMPARE ERROR : die Speicherkonfigurationsregister zeigen keine signifikanten Fehler an, aber das erwartete Bitmuster ist dennoch ungleich dem gelesenem Bitmuster

UNDEFINED ERROR : vom Hauptspeicher wurde ein Fehler erkannt, dessen Ursache jedoch vom Programm nicht ermittelt werden konnte; Aus diesem Grund sind weitergehende Analysen zur Fehlerortung nur durch den Bediener möglich.

Um eine Fehleranalyse bis möglichst auf den bzw. die fehlerhaften Schaltkreise genau realisieren zu koennen, werden vom Programm PVKAS programminterne Informationen und die Inhalte der Speicherkonfigurationsregister ausgewertet.
Im Ergebnis dessen, wird der Fehlertyp (s.o.) durch folgende Informationen vervollständigt:

- der Kontroller, welcher den Fehler anzeigte (LOWER CONTROLLER bzw. UPPER CONTROLLER),
- die fehlerbehafte MEM-Matrix (May : xx (Slot xx)),
- die fehlerbehaftete Bank (Bank : xx),
- das aktuelle Fehlersyndrom (Error Syndrom : xx),
- die fehlerhafte(n) Bitstelle(n) und den bzw. die fehlerhaften Schaltkreise (Bit : Dxx ---> Chip Dxxx)

ausgeschrieben.

Alle angegebenen Fehlertypen, außer COMPARE ERROR, können gemeinsam mit dem Fehler HIGH ERROR RATE auftreten und führen z.B. zur Mitteilung

LOWER CONTROLLER: SINGLE BIT ERROR / HIGH ERROR RATE

HIGH ERROR RATE zeigt an, daß vor Beendigung der Behandlung eines erkannten Fehlers mindestens ein weiterer Fehler aufgetreten ist.
Die nächsten Zeilen der Fehlermitteilung geben den erwarteten u. den gelesenen Inhalt der aktuell untersuchten Speicherzelle sowie den XOR-Vergleich des erwarteten und des gelesenen Inhaltes einschließlich der dabei als nicht übereinstimmend erkannten Bitstellen an. Diese Angabe erscheint jedoch sinnvollerweise nur beim Fehlertyp COMPARE ERROR.
Zur effektiven Fehlerortung wird anschließend die Wertebelegung der Speicherkonfigurationsregister ausgegeben, die vom Programm im Moment des Erkennens eines vorliegenden Speicherfehlers vorgefunden worden ist. Innerhalb der eckigen Klammern ist die physische Adresse des jeweiligen Speicherkonfigurationsregisters (abhängig vom eingestellten TR-level des Hauptspeichers) angegeben.

Fehler anderer Natur, wie z.B. fehlerhaft konfigurierter Hauptspeicher, werden vom Diagnoseprogramm ebenfalls erfaßt und durch selbterklärende Fehlermitteilungen angezeigt. Bei derartigen Fehlern wird generell in den Haltzustand übergegangen. Mit entsprechenden Konsolkommandos sind dann weitere Bedienerhandlungen möglich. Es ist zu beachten daß bei Neukonfigurierung des Hauptspeichers (insbesondere beim Übergang vom überlagerten Betrieb in den nicht überlagerten Betrieb und umgekehrt) das Programm PVKAS neu zu laden und zu starten ist.

6.5.6. Aktionen bei unerwarteten Fehlern

Interrupts und Ausnahmen, die nicht vom Diagnoseprogramm initiiert werden, entstehen als Folge von unerwarteten Fehlern und führen zur Ausgabe folgender Zeilen:

```
Unexpected trap/int/exc at SCB Vector xx, return PC would be xxxxxxxx
  HALT INST EXECUTED
  HALTED AT xxxxxxxx
```

Wurde ein Interrupt über den Vektor %X60 (memory write timeout) oder den Vektor %X04 (machine check) erkannt, folgt eine dementsprechende selbsterklärende Mitteilung (mit Ausnahme der Interrupts und Ausnahmen, die programmbedingt während der Fehleranalyse und -lokalisierung auftreten und programmintern behandelt werden).
Bei Auftreten der unerwarteten Fälle wird daran anschließend in den Haltzustand übergegangen. Gegebenenfalls ist dann mit dem Konsolkommando >>>C die Fortführung des Grundspeichertests möglich.

6.5.7. Programmitteilungen

Nach jedem Durchlauf aller abzuarbeitenden Tests des Grundspeichertests erfolgt die Programmitteilung:

```
PVKAS VX.X pass # x(X) done at time xx:xx:xx.xx
```

Wurde während des letzten Passes mindestens ein Fehler festgestellt, so wird daraufhin eine Fehlertabelle ausgegeben. Diese Tabelle hat folgenden Aufbau:

ERROR TYPE	ERROR NUMBER (HEXADECIMAL) LAST PASS	SUMMARY
COMPARE ERROR	xxxxxxxx	xxxxxxxx
SINGLE BIT ERROR	xxxxxxxx	xxxxxxxx
MULTIPLE BIT ERROR	xxxxxxxx	xxxxxxxx
SING & MULT BIT ERROR	xxxxxxxx	xxxxxxxx
UNDEFINED ERROR	xxxxxxxx	xxxxxxxx
SINGLE BIT ERROR / HIGH ERROR RATE	xxxxxxxx	xxxxxxxx
MULTIPLE BIT ERROR / HIGH ERROR RATE	xxxxxxxx	xxxxxxxx
SING & MULT BIT ERROR / HIGH ERROR RATE	xxxxxxxx	xxxxxxxx
UNDEFINED / HIGH ERROR RATE	xxxxxxxx	xxxxxxxx

Programmmitteilungen, welche den Testablauf wiedergeben, sind ebenfalls möglich. Die Ausgabe derartiger Ausschriften kann vom Bediener durch entsprechende Wertebelegung des programmspezifischen Steuerregisters CMDREG 5 zugelassen oder unterdrückt werden (s. Abschn. 6.5.9.). Die Ausschrift

```
Test x
  Subtest y
```

teilt z.B. mit, daß aktuell der Test x abgearbeitet wird. Mit

```
WAIT   bzw. -------------------- END OF WAIT -----
```

wird dagegen der Ablauf der Subtests innerhalb des Tests 3 protokolliert.

6.5.8. Programmaufbau

Der Grundspeichertest ist so angelegt, daß

- die Funktionsfähigkeit des Hauptspeichers umfassend geprüft wird,
- der Testablauf in einfacher Weise durch den Bediener gesteuert werden kann und
- mittels relevanter, aussagekräftiger Fehlermitteilungen dem Bediener eine größtmögliche Unterstützung bei der Fehlerortung gegeben wird.

Die Funktionsfähigkeit des Hauptspeichers wird zum einen auf der Grundlage der im Abschn. 6.6. beschriebenen Tests und Subtests und zum anderen durch umfassende Auswertung der in den Speicherkonfigurationsregistern bereitgestellten Informationen geprüft. Solche Informationen sind u.a. relevante Fehlermitteilungen und Hauptspeicherzustandsinformationen

Der konkrete Testablauf (s. Abschn. 6.5.9.) wird durch programmspezifische Steuerregister (command register - CMDREG x) festgelegt. Mittels der im Abschn. 6.5.9. beschriebenen Bedienerkommunikation kann der Inhalt dieser Steuerregister sowohl statisch (vor Beginn der eigentlichen Tests) festgelegt als auch dynamisch (während des laufenden Grundspeichertests) variiert werden. Nach dem Laden und dem erstmaligem Start des Grundspeichertests werden vom Diagnoseprogramm Standardwerte vorgegeben.

Nachfolgend ist die Speicherplatzaufteilung beim Diagnoseprogramm PVKAS angegeben:

loadaddr +	Inhalt
22C	Übergabeparameter zwischen Bediener und Programm
248	Variable, Tabellen
41D	Fehlersyndrometabellen
49F	Tabelle der Schaltkreisnamen
4D6	Prüfdatenmuster
800	Kernel-Stack
A00	Interrupt-Stack
C00	Systemsteuerblock (system control block - SCB)
D00	Standardtexte
122F	Fehlermitteilungen
1D59	Programminitialisierung
227B	Tests 1 bis 4
2E94	Routinen zur Exception- und Interruptbehandlung
3282	Unterprogramme
4600	erste Speicherzelle nach dem Diagnoseprogramm (gleichzeitig Beginn des zweiten Programmbereiches zum Umlagern des Diagnoseprogrammes)

6.5.9. Bedienerkommunikation und Testablaufsteuerung

Mit Hilfe eines Dialoges von Bediener und Programm können in einfacher Weise die Testabarbeitungsbedingungen und der Testablauf selbst variiert werden.
Die Aufnahme eines Dialoges ist durch

- Start des Grundspeichertests mit dem Kommando 'S 204' oder
- Eingabe von CTRL/C (^C) während des Programmlaufes

möglich.

Um einen korrekten Testablauf sichern zu können, wird vom Programm nur zu bestimmten, dem Testablauf entsprechend sinnvollen Zeitpunkten auf eine Eingabe von CTRL/C reagiert. Infolgedessen können u.U. lange Wartezeiten (insbesondere während des Tests 2) für den Bediener enstehen.
Das Programm meldet sich mit der Ausgabe des aktuellen Inhaltes der programmspezifischen Steuerregister CMDREG x (s. Abschn. 6.4.) und der Zeile

INPUT CMDREG (0|...|6|<CR>|^C) :

Der Bediener kann daraufhin durch Eingabe eines Kommandos vom Diagnoseprogramm die Ausführung spezieller Funktionen abfordern.

Kommando	Wirkung
<CR>	Zeigt an, daß keine Änderung des Inhalts der Steuerregister CMDREG x sondern der sofortige Start bzw. die Fortführung des Grundspeichertests gefordert wird.
^C	Bewirkt die Ausgabe des aktuellen Inhalts der Steuerregister CMDREG x (s.a. Abschn. 6.4.).
0|1| ... |6	Spezifiziert das Steuerregister, dessen Inhalt geändert werden soll. Falls der Umlagerungsmodus bereits festgelegt ist, läßt das Programmm das Ändern von CMDREG 0 oder CMDREG 1 nicht zu.

Falls die Änderung des Inhalts eines Steuerregisters gewünscht wird, antwortet das Diagnoseprogramm mit der Ausgabe der Zeile

Input new value (<CR>|^Z) : 0

Nunmehr kann der Bediener die Aktualisierung des Registerinhaltes vornehmen, indem er den gewünschten Hexadezimalwert eingibt und die Eingabe mit <CR> abschließt. Prinzipiell werden von rechts nach links 8 Ziffern als Eingabewert akzeptiert. Mit ^Z wird eine Eingabe für ungültig erklärt. Mittels Eingabe von <DEL> wird die jeweils zuletzt eingegebene Ziffer gelöscht.

Das Diagnoseprogramm bestätigt den Erhalt eines Hexadezimalwertes, indem es den aktuellen Inhalt des zu bearbeitenden Steuerregisters und den eingegebenen Hexadezimalwert angibt. Daran anschließend werden nichtsignifikante Bitstellen des Eingabewertes gelöscht bzw. Korrekturen vorgenommen, um eine hohe Sicherheit bei der Ausführung des Tests gewährleisten zu können. Der erzeugte Wert wird in spezielle Programmvariable übernommen und ebenfalls entsprechend dem nachfolgenden Format ausgegeben.

alter Wert ---> Eingabewert ---> eingetragener Wert

Daraufhin wird die Bedienerkommunikation mit der Ausgabe der Zeile

INPUT CMDREG (0| ... |6|<CR>|^C) :

nach dem oben beschriebenen Algorithmus fortgeführt.
Vor Ausführung eines Tests bzw. bestimmter Subtests wird geprüft, ob aktuell im Umlagerungsmodus oder im Modus ohne Umlagerung gearbeitet wird. Davon abhängig werden die vom Bediener vorgegebenen Testbereichsgrenzen überprüft und ggf.

zugeordnete Fehlermeldungen ausgegeben.

Nachfolgend ist die Semantik der Steuerregister CMDREG x angegeben.

CMDREG 0 - Enthält die Anfangsadresse des zu prüfenden Hauptspeicherbereiches. Der eingetragene Wert wird vom Diagnoseprogramm ggf. selbsttätig auf eine Quadwortgrenze korrigiert.
Standard: Nach Programmstart mittels >>>S 200: 0
Nach Programmstart mittels >>>S 204: %X4600

CMDREG 1 Enthält die um eins erhöhte Endeadresse des zu prüfenden Hauptspeicherbereiches; Der eingetragene Wert wird vom Diagnoseprogramm ggf. selbsttätig auf eine Quadwortgrenze korrigiert.
Standard: Nach Programmstart mittels >>>S 200: Adresse der ersten nichtexistierenden Speicherzelle
Nach Programmstart mittels >>>S 204: %X200000

CMDREG 2 - Gibt an, welche Tests ausgeführt bzw. nicht ausgeführt werden sollen:
BIT<0> = 1 ; Test 1 ausführen
BIT<1> = 1 ; Test 2 ausführen
BIT<2> = 1 ; Test 3 ausführen
BIT<3> = 1 ; Test 4 ausführen
Standard: %X7

CMDREG 3 - Gibt an, ob der Grundspeichertest
als Dauertest: alle Tests werden zyklisch beliebig lange wiederholt
oder

als Einzeltest: die ausgewählten Tests (s. Wertebelegung CMDREG 2) werden nur einmal abgearbeitet

ausgeführt werden soll.
BIT<0> = 1 ; Dauertest
= 0 ; definierte Anzahl von Durchläufen (Pässen)
BIT<31:16> ; enthält als Dualzahl die Anzahl der auszuführenden Pässe
Standard: %X10000

CMDREG 4 - Legt fest, ob nach Auftreten eines Fehlers in den Haltzustand übergegangen werden soll:
BIT<0> = 1 ; Halt nach Fehler gefordert
= 0 ; kein Halt nach Fehler
Standard: 1

CMDREG 5 - Bestimmt den Umfang an Fehlermitteilungen und Testausschriften:
BIT<0> = 1 ; Fehlermitteilungen sind auszugeben
= 0 ; Fehlermitteilungen sind zu unterdrücken
BIT<1> = 1 ; ausführliche Fehlermitteilung gefordert
= 0 ; gekürzte Fehlermitteilung gefordert
Dieses Bit wird nur bei BIT<0> = 1 ausgewertet.

BIT<2> = 1 ; Signal bei Fehler gefordert
= 0 ; kein Signal bei Fehler
BIT<3> ; reserviert
BIT<4> = 1 ; Testausschriften gefordert
= 0 ; Testausschriften sind zu unterdrücken

BIT<31:16> ; Dualzahl, die festlegt, nach welcher Anzahl von erkannten SINGLE BIT ERROR keine Fehlerausgabe bei weiteren Fehlern dieses Typs mehr erfolgen soll. (Nur wirksam falls BIT<0> = 1; ist für alle Tests jeweils neu gültig.)
Standard: %XA0013

CMDREG 6 - Legt fest, ob nach Auftreten eines Fehlers die fehlerhafte Speicherzelle wiederholt gelesen werden soll:
BIT<0> = 1 ; Fehlerschleife ausführen
= 0 ; keine Fehlerschleife gefordert
Standard: 0

6.6. Testbeschreibung

6.6.1. Test 1 - Basistest

Der Test 1 ist ein einfacher Schreib-/Lesetest. Hierbei werden in einem ersten Schritt alle zu prüfenden Speicherzellen mit einem festen Prüfmuster quadwortweise beschrieben und in einem zweiten Schritt quadwortweise ausgelesen und auf korrekten Inhalt hin überprüft.

Subtest 1.1.	Prüfmuster %X0000 0000 FFFF FFFF
Subtest 1.2.	Prüfmuster %XFFFF FFFF 0000 0000

6.6.2. Test 2 - Marchtest

Der Test 2 realisiert einen Marching-Algorithmus. Zunächst wird dazu der zu prüfende Speicherbereich mit dem Prüfmuster %XA55A A55A langwortweise beschrieben.
Im Subtest 2.1 wird anschließend der zu prüfende Speicherbereich langwortweise behandelt, indem jeweils ein Langwort gelesen, mit dem Vorgabewert verglichen und sofort mit dem invertiertem Wert %X5AA5 5AA5 überschrieben wird.
Im Subtest 2.2 wird analog zum Subtest 2.1 vorgegangen. Ausgangswert ist jedoch nunmehr %X5AA5 5AA5 und überschrieben wird mit %X9494 9494.
In den Subtests 2.3 und 2.4 wird der Hauptspeicher byteweise behandelt.
Im Subtest 2.3 wird vom Anfang bis zum Ende des Prüfbereiches jedes Byte gelesen, geprüft (Vorgabewert ist %X94) und sofort mit dem invertiertem Wert %X6B überschrieben.
Der Subtest 2.4 läuft wieder analog wie der Subtest 2.3 ab. Vorgabemuster ist jedoch %X6B und überschrieben wird mit %X5A.

6.6.3. Test 3 - Refreshtest

Der Test 3 untergliedert sich in die Subtests 3.1 bis 3.9. Alle diese Subtests sind nach der folgenden Prüfstrategie organisiert:

Hintergrundschreiben : Der zu prüfende Speicherbereich wird mit einem definiertem Bitmuster beschrieben.

Warten : Nach dem Beschreiben unterbleiben für eine Wartezeit von ca. 5 sec jegliche Zugriffe zum Prüfbereich des Speichers.

Lesen Danach wird der Prüfbereich gelesen und der Dateninhalt geprüft.

Die einzelnen Subtests unterscheiden sich lediglich dadurch, das unterschiedliche Bitmuster als Hintergrundmuster genutzt werden. Im Subtest 3.1 wird von der Anfangsadresse bis zur Endeadresse des zu prüfenden Speicherbereiches das Schiebemuster %X0000 0001 langwortweise eingeschrieben, d.h. eine durchlaufende "1" ab Bit 0 realisiert (also Schreiben von %X0000 0001, %X0000 0002, %X0000 0004 usw. auf fortlaufenden Langwortadressen).
Im Subtest 3.2 gelangt stattdessen das Schiebemuster %XFFFF FFFE zur Anwendung (also Schreiben von %XFFFF FFFE, %XFFFF FFFD, %XFFFF FFFB usw. auf fortlaufenden Langwortadressen).
In den Subtests 3.3 bis 3.9 werden anstelle der Schiebemuster feste Prüfmuster genutzt, die in der folgenden Übersicht aufgeführt sind:

Subtest	3.3	Prüfmuster	%X0500 0000
"	3.4	"	%X0001 0000
"	3.5	"	%X0004 0009
"	3.6	"	%X1202 0000
"	3.7	"	%X0080 0000
"	3.8	"	%X0000 2000
"	3.9	"	%X1000 0100

6.6.4. Test 4 - Netzausfalltest

Der Test 4 prüft, ob bei einem Netzausfall mit Hilfe des Batteriestützmoduls der aktuelle Speicherinhalt über mindestens 10 min aufrechterhalten wird.
Zu diesem Zweck werden:

a) der aktuelle Prüfbereich mit dem Schiebemuster %XFFFF FFFE beschrieben und eine Prüfsumme über die ersten 31 Langwörter der Restartroutine innerhalb des Test 4 gebildet,

b) mittels entsprechender Bildschirmausschriften der Bediener zu bestimmten Bedienerhandlungen aufgefordert:

- Einlegen der Konsoldiskette DS1
- Unterbrechen der Stromversorgung
- Wiedereinschalten nach 10 min
- Neustart mittels Konsolkommando >>>S 208 (bzw. loadaddr+208)

c) nach Neustart des Programmes die ersten 31 Langwörter der Restartroutine mittels Prüfsummenbildung abgeprüft und bei Korrektheit daran anschließend der Inhalt des Testbereich untersucht.

Die korrekte Durchführung dieses Tests erfordert:

- daß der AUTO-RESTART-Schalter auf "OFF" geschaltet ist
- zum Netzaus- und -einschalten der Netzhauptschalter (nicht der Kippschalter des Stromversorgungsmoduls des Speichersubsystem) genutzt wird.

6.7. Auflistung der Tests

Test 1: Basistest
Test 2: Marchtest
Test 3: Refreshtest
Test 4: Netzausfalltest

6.8. Abkürzungen

CMDREG - command register (programmspezifische Steuerregister)

7. PVKAB Architekturbefehlstest

7.1. Zusammenfassung

Das Programm PVKAB (Dok.Nr.:1.57.00003.4) ist ein Programm der Ebene 2. Es ist für den Test aller Architekturbefehle des RVS -Rechnersystems Robotron K1840 bestimmt. Die Gleitkommabefehle werden von diesem Programm nicht überprüft. Dies erfolgt im Programm PVKAC. Das Programm PVKAB ist unter dem Diagnosesupervisor im Stand-alone-Modus oder als Nutzertask unter SVP lauffähig.

7.2. Testbedingungen

7.2.1. Hardware

Dieses Programmm kann auf beliebiger K1840-Rechnerkonfiguration abgearbeitet werden.

7.2.2. Software

Diagnosesupervisor

7.3. Voraussetzungen

Mikrodiagnose und PVKAA müssen erfolgreich gelaufen sein.

7.4. Bedienungshinweise

7.4.1. Laden und Starten

Nach dem Laden des Programmes sind folgende Bedienhandlungen auszuführen:

```
DS> ATT KAA40 ISB KAn (1) (2) (3) (4)
    (1)  G - Floating Befehle (n)
    (2)  H - Floating Befehle (n)
    (3)  WCS letzte Adresse
    (4)  Gleitkommabeschleuniger (FPA) (0 = FPA nicht vorh. 1 = FPA vorh.)

    DS> SEL ALL
    DS> ST
```

7.4.2. Optionen

Optionen werden durch ATTACH-Kommandos festgelegt [Gleitkommabeschleuniger (FPA)].

7.4.3. Testweiterführung

Innerhalb RITS existiert die Möglichkeit, durch Fehler und (oder) Interrupts unterbrochene Programme weiterzuführen. Speicherverwaltungszugriffsverletzungen führen zusätzlich zu den spezifischen Fehlern und Traps zu Fehlerunterbrechungen. Die Testweiterführung kann über die Steuer- und Ereignisflags gesteuert werden. Ist das QUICK-Flag gesetzt, so ist die Testweiterführung nicht möglich. Bei gesetztem Ereignisflag 2 ist der Intervalltimerinterrupt gesperrt, aber der Speicherzugriffsverletzungs-Fault ist möglich (wenn Quick-Flag=0).
Es ist zu beachten, daß dieses feste Interrupts sind. Eine Ausnahme besteht dann, wenn die Diagnosesupervisor-Clock-Routine verboten wird. Bei gesetztem Ereignisflag 3 (Flags 2 und Quick = 0) kommt der Interrupt zur gleichen Zeit wie der Speicherverwaltungszugriffsverletzungs-Fault.
Unter SVP kann der Intervalltimer nicht gesteuert werden. In diesem Fall haben Flag 2 und 3 keinen Effekt.

7.5. Funktionsbeschreibung

7.5.1. Übersicht

7.5.1.1. Befehlstest-Service-Routine (RITS)

Diese Routine wird bei jedem Test durchlaufen um die Testdatentabelle zu dekodieren, die Testfälle im Speicher zu verankern, die Befehle auszuführen und Fehler zu erkennen.
Nach dem Festlegen der Zähler für den jeweiligen Test wird das Dekodierprogramm RIT_DECODE (PVKAB3) aufgerufen, um alle Testdatenpuffer mit den Testdaten zu initialisieren. Danach durchläuft RITS die Eintrittspunkttabelle, um die Befehlskette zu bilden, ruft für jeden Operanden RITS_OPERANDS auf und läßt durch RIT_EXECUTE (PVKAB5) die Befehle ausführen. Das Exception wird entweder vom Testbefehl oder von der nächsten Anweisung erzwungen (BPT), falls dieser den Übergang zu RIT_EXCEPTION verlegt. Damit wird der Maschinenstatus gerettet und die Steuerung wieder RITS übergeben. Zum Schluß wird, wenn alle Daten geprüft und ein Fehler erkannt wurde, über den Diagnosesupervisor, der dann seinerseits RIT_ERROR (PVKAB7) aufruft, die normale und die erweiterte Fehlerinformation angezeigt.

7.5.2. Programmgröße

56832 Bytes
111 Pages (Speicherseiten)

7.5.3. Programmlaufzeit

3:40 min

7.5.4. Abarbeitungsbedingungen

entfällt

7.5.5. Flagverwendung

In diesem Test sind die Ereignisflags 2 bis 6 aktiv.

Flag 2: Verbietet den Intervalltimerinterrupt während der Befehlsabarbeitung.
Flag 3: Erlaubt den Intervalltimerinterrupt, wenn Page-Faulting auch erlaubt ist.
Flag 4: Erlaubt Fortführung eines Subtests nach Fehler (normalerweise wird dieser abgebrochen).
Flag 5: Verbietet Ausführung des Befehls DIVP während Intervalltimerinterrupt.
Flag 6: Erlaubt dem Nutzer eine eigene Testfolge zusammenzustellen, nachdem angefragt wurde, welche Tests abgearbeitet werden sollen. Ist dieses Flag gesetzt, fordert das Diagnoseprogramm die Nummern der auszuführenden Tests an. Diese werden einzeln eingegeben. Ist diese eingegeben, so drücken Sie <CR> statt der Testnummer und das Programm wird abgearbeitet. Es kann irgendeine der Testnummern eingeben werden. Falls Sie eine Teilsektion aufrufen und die Testnummer ist nicht in dieser Sektion, wird der Test nicht ausgeführt (d.h. die Sektion hat höherePriorität als die Anwahl des Flag6).

** Dieses Flag arbeitet nicht, wenn das Operator-Flag gelöscht ist.

Anmerkung: Um die Abarbeitungszeit und den Aufwand zu verringern, werden die Flags nur am Beginn jedes Tests oder nach einem Fehler abgefragt. Deshalb verändert ein Drücken von "^C" und Setzen der Ereignisflags das laufende Programm nicht (außer im Fehlerfalle).

7.5.6. Allgemeine Programmanforderungen

entfällt

7.5.7. Fehlerermittlung

Der Diagnosesupervisor erzeugt Standardausschriften, welche die Programmidentifikation, die Testnummer, die Subtestnummer, die Fehlernummer, die Fehlerart, das zu testende Gerät (KA0) und eine von RITS erzeugte Ausschrift, in welcher die fehlerhafte Funktion zu finden ist, enthalten.
Die Fehlernummer ist die Adreßmoduseintrittsnummer in der Tabelle (RAM)+1000, d.h. Nr.1005 besagt, daß die 6. Kombination (0 ist die erste) nach Ausführung verlassen wurde, als der Page-Faulting-Interrupt die Ausführung der Anweisungen unterbrochen hat.

Beispiel:

```
******* PVKAB,RVS Basic Instructions Excerciser *******

Pass 1   Test 1   Subtest 3   Error 0    03-NOV-1987  14:23:09.43

HARD ERROR WHILE TESTING KA0   Operand data error
```

7.5.7.1. Grundfehlerausschriften

Diese Fehlerausschrift enthält den Testbefehl in Macro-Schreibweise einschließlich der Operanden.
Geschah der Fehler während Page-Fault, dann wird dies angezeigt. Trat eine falsche Exception ein, wird der erwartete und der aktuelle Wert angegeben. Am Schluß werden alle Operanden und Register, welche fehlerhafte Daten enthalten, ausgegeben.
Die Operanden werden als "OPRn" angegeben, wobei n eine Zahl zwischen 1 und 6 ist. Falls der Operand eine Adresse bezeichnet, wird diese auf derselben Zeile ausgegeben. Die nächste Zeile enthält dann die Daten (bezeichnet als "@OPR"). Einige Operanden verweisen auf implizite Operanden. Diese werden in eckige Klammern gesetzt. Beispiel: "[OPR3]"

Beispiel:

```
Instruction:  INSQTI    (R2),(R3)
    with page faults

expected exceptions:    Reserved operand data
actual   exception :    None

       ADDRESS     INITIAL       EXPECTED    ACTUAL      XOR
OPR2   00044DF8
@OPR   00044DF8    00000008      00000008    00000009    00000001
OPR4   00044E17    FFFFFFE1      FFFFFFE1    FFFFFFF1    00000010
Registers:

       PC          000449FF      000449FF    00044A02    000003FD
```

7.5.7.2. Erweiterte Fehlerausschriften

Sind erweiterte Fehlerausschriften erlaubt, werden alle wichtigen Informationen über den Testbefehl und die nicht exakten Daten mit Differenzen angegeben.
Trat eine Exception auf, so wird dies angezeigt (außer BPT). Alle Operanden und alle genutzten Register werden angegeben.

Beispiel:

```
Instruction:  CVTLB    (R9),R10
   with page faults

Exception  : T - bit Trap

       ADDRESS    INITIAL     EXPECTED    ACTUAL      XOR
OPR1   000465FF   A0000000    A0000000    A0000000
OPR2   R10              5A          00          90       90
Register:

       R9         000465FF    000465FF    000465FF
       R10        5A5A5A5A    5A5A5A00    5A5A5A90    00000090
       PC         000461FF    00046202    00046202
       PSL        03C000D2    03C000DA    03C000D6    0000000C
```

7.5.8. Programmitteilungen

entfällt

7.5.9. Leistungsparameter des Testobjektes

entfällt

7.5.10. Aktionen bei unerwarteten Fehlern

Werden aus verschiedenen Gründen vom Testprogramm Datenfehler in einer der Testtabellen gefunden oder wenn das Testprogramm aus irgendwelchen Gründen nicht mehr "weiter weiß", wird durch die Auschrift "FATAL ERROR" angezeigt, daß ein unvorhersehbarer Fehler das Programm verändert hat und deshalb ein weiterer Test unmöglich ist. Es wird "FATAL ERROR" angezeigt und das Programm abgebrochen. Fehler dieses Typs haben Fehlernummern zwischen 200 und 999, wobei die Hunderter den Programmteil bezeichnen, der den Fehler aufgerufen hat, d.h. Fehler 301 kommt vom PVKAB3.

7.5.11. Testfolge bei mehreren Einheiten

enfällt

7.5.12. Einstellungen für Ablauf unter AFT

entfällt

7.5.13. Testsektionen

Beim PVKAB-Testprogramm existieren mehrere Testsektionen, die nachfolgend mit den entsprechend dazugehörigen Befehlen aufgeführt sind.

INTEGER

arithmetische und logische Integerbefehle:

MOVB , MOVW , MOVL , MOVQ , CLRB , CLRW , CLRL , CLRQ ,
MNEGB , MNEGW , MNEGL , MCOMB , MCOMW , MCOML , MOVZBW , MOVZBL ,
MOVZWL , CVTBW , CVTBL , CVTWB , CVTWL , CVTLB , CVTLW , CMPB ,
CMPW , CMPL , TSTB , TSTW , TSTL , ADDB2 , ADDB3 , ADDW2 ,
ADDW3 , ADDL2 , ADDL3 , ADAWI , INCB , INCW , INCL , ADWC ,
SUBB2 , SUBB3 , SUBW2 , SUBW3 , SUBL2 , SUBL3 , SBWC , MULB2 ,
MULB3 , MULLW2 , MULW3 , MULL2 , MULL3 , EMUL , DIVB2 , DIVB3 ,
DIVW2 , DIVW3 , DIVL2 , DIVL3 , EDIV , ASHL , ASHQ , BITB ,
BITW , BITL , BISB2 , BISB3 , BISW2 , BISW3 , BISL2 , BISL3 ,
BICB2 , BICB3 , BICW2 , BICW3 , BICL2 , BICL3 , XORB2 , XORB3 ,
XORW2 , XORW3 , XORL2 , XORL3 u. ROTL

arithmetische Integer-Steuerbefehle

ACBB , ACBW , ACBL , AOBLEQ , AOBLSS , SOBGEQ u. SOBGTR

FIELD

Befehle mit variabler Länge des Bitfeldes

EXTV , INSV , CMPV , CMPZV , FFC u. FFS

Einzelbitfeldsteuerbefehle

BBCS , BBSC , BBCC , BBSSI , BBCCI , BLBS , BLBC , BBS
BBC u. BBSS

CONTROL

Grundsteuerbefehle

BRB , BRW , BNEQ , BEQL , BGTR , BLEQ , BGEQ , BLSS
BGTRU , BLEQU , BVC , BVS , BGEQU u. BLSSU

Einzelbitsteuerbefehle

BBS , BBC , BBSS , BBCS , BBSC , BBCC , BBSSI , BBCCI ,
BLBS u. BLBC

arithmetische Integer-Steuerbefehle

ACBB , ACBW , ACBL , AOBLEQ , AOBLSS , SOBGEQ u. SOBGTR

QUEUE

Warteschlangenbefehle

INSQUE , REMQUE , INSQHI , INSQTI , REMQHI u. REMQTI

CHARACTER

Zeichenkettenbefehle"

MOVC3 , MOVC5 , MOVTC , MOVTUC , CMPC , SCANC , SPANC , LOCC
SKPC u. MATCHC

DECIMAL

Dezimal-Zeichenkettenbefehle

MOVP , CMPP3 , CMPP4 , ADDP4 , ADDP6 , SUBP4 , SUBP6 , MULP ,
MULP , DIVP , CVTLP , CVTPL , CVTPT , CVTTP , CVTPS ,CVTSP u.
ASHP

MOVXMNEGX

alle MOVX , MOVCn ,MCOMX , MOVTXX , MNEG Befehle

MOCB , MOVW , MOVL , MOVQ , MNEGB , MNEGW , MNEGL , MCOMB ,
MCOMW , MCOML , MOVZBW , MOVZBL , MOVZWL , MOVC3 , MOVC5 , MOVTC ,
MOVTUC u.MOVP

CMPXX
---

alle CMPX und CMPXX Befehle

CMPB , CMPW , CMPL , CMPP3 , CMPP4 , CMPC3 , CMPC5 , CMPV u.
CMPZV

CVTXX
---

alle CVTXX Befehle

CVTLP , CVTPL , CVTPT , CVTTP , CVTPS u. CVTSP

ADDMULSUBDIV
---

alle ADDxx , SUBxx , MULxx und DIVxx Befehle

ADDB2 , ADDB3 , ADDW2 , ADDW3 , ADDL2 , ADDL3 , SUBB2 , SUBB3 ,
SUBW2 , SUBW3 , SUBL2 , SUBL3 , MULB2 , MULB3 , MULW2 , MULW3 ,
MULL2 , MULL3 , DIVB2 , DIVB3 , DIVW2 , DIVW3 , DIVL2 u. DIVL3

ASHX
---

alle ASHx Befehle

ASHL , ASHQ u. ASHP

7.5.14. Verschiedenes

7.5.14.1. Übergehen eines Testfalles

Sollte es notwendig sein, einen Test zu umgehen, weil auf AEM oder aus anderen Gründen gewartet wird, so kann die folgende Prozedur durchgeführt werden.

(1) Laden des Programmes.
(2) Es ist die Basisadresse der P_SECT des Tests (z.B.TEST002) im Listing dieses Programmeszu suchen.
(3) Setzen des Basisregisters mit diesem Wert.
(4) Finden der "TT"-Zeile des richtigen Testfalles.
(5) Zählen der Bytes in der Expansion des Testfalles und davon 3 subtrahieren.
(6) Zusammenschreiben der Adresse aus Schritt 4 mit dem Wert aus Schritt 5 in ein Langwort ab 03FC.
(7)Rücksetzen des Relocation-Registers zurück auf 0.

7.6. Testbeschreibung

Im Architekturbefehlstest gibt es folgende Programmodule:

PVKAB1 - Header Modul
PVKAB2 - RITS (Instruction Test Services)
PVKAB3 - RIT Decode
PVKAB4 - RIT Operands
PVKAB5 - RIT Execute
PVKAB6 - RIT Exception
PVKAB7 - Error Reporting Routine
PVKAB8 - Test Instruction Decode Dispatch Tables
PVKAB9 - Global Test Data Pools
PVKABA - Integerbefehle
PVKABB - Variable Befehle
PVKABC - Steuerbefehle
PVKABD - Indexbefehl
PVKABE - Warteschlangenbefehle
PVKABF - Zeichenkettenbefehle
PVKABG - Gepackte-Dezimalzahlbefehle
PVKABH - EDITPC-Befehl

7.6.1. Modulbeschreibung

Jedes der oben angegebenen Module enthält sämtliche Informationen über die jedem der Befehlstestmodule zugeordneten Befehle, die zur Dekodierung eines Testtabelleneintrags und zur Ausführung der Befehle benötigt werden. Die zu dekodierende Dispatchliste wird von RITS-Decode genutzt. Das erste Byte der Liste gibt die Länge der Liste an. Das zweite Byte gibt die Länge des Befehls in Byte an. Die restlichen Bytes sind "Hauptstatus"-Indexe die zur Steuerung von RITS-DECODE bei dessen Interpretation der Testtabelleneinträge genutzt werden. Diese Indexe werden durch im Speicher stehende Operanden aufgerufen und sie schreiben den Befehl in den Speicher. Weiterhin kellern sie die Rücksprungadresse ein, geben die Befehlsadresse im Stack an und übernehmen die Ausführung eines REI. (Beim PVKABE folgt der zu dekodierenden Dispatchliste eine ASCIC-Zeichenkette die den Befehlsmnemonik enthält. Dies wird bei der Fehlerreporting-Routine genutzt).
Der Zugriff zum Feld, welches die obigen Informationen enthält , wird mittels der Tabelle "RID OFFS. TAB", welche Teil des RITS- Moduls ist, getan. Die Tabelle wird von einem Operationscode indexiert und dessen Wortinhalt wird zur Adresse "RID AREA" addiert. Dies geschieht, um die Adresse der Dekodier-Dispatchliste für einen Testbefehl zu bilden.
Die jeweiligen einzelnen 161 Tests überprüfen die Operationen des Microcodes für die zu testenden Befehle, und die Befehlszwischenspeicherung und die Bedingungscodeslogik. Die beiden letztgenannten Funktionen sind bei Befehlen gleicher Art gleich.
Jeder Test besteht aus mehreren Subtests. Diese einzelnen Subtests unterscheiden sich dadurch, daß in ihnen jeweils verschiedene Operanden und Adreßmodi genutzt werden.

7.7. Abkürzungsverzeichnis

BPT	Break-Point-Trap
SVP	Virtuelles Betriebssystem der RVS K1800-Systeme
AEM	Änderungsmeldung
WCS	ladbarer Mikroprogrammspeicher
RAM	Adreßmodus
RIT	Robotron Befehlstest
RITS	Robotron Befehlstestservices
RID	Robotron K1800 Befehlstestidentifikation

7.8. Auflistung der Tests

7.8.1. Die Tests des PVKABA

TEST	BEFEHL	TEST	BEFEHL	TEST	BEFEHL
1	BRB	2	BRW	3	BBC
4	BBS	5	MOVB	6	MOV
7	MOVL	8	MOVQ	9	CLRB
10	CLRW	11	CLRL	12	CLRQ
13	MNEGB	14	MNEGW	15	MNEGL
16	MCOMB	17	MCOMW	18	MCOML
19	MOVZBW	20	MOVZBL	21	MOVZWL
22	CVTBW	23	CVTBL	24	CVTWB
25	CVTWL	26	CVTLB	27	CVTLW
28	CMPB	29	CMPW	30	CMPL
31	TSTB	32	TSTW	33	TSTL
34	ADDB2	35	ADDB3	36	ADDW2
37	ADDW3	38	ADDL2	39	ADDL3
40	ADAWI	41	INCB	42	INCW
43	INCL	44	ADWC	45	SUBB2
46	SUBB3	47	SUBW2	48	SUBW3
49	SUBL2	50	SUBL3	51	DECB
52	DECW	53	DECL	54	SBWC
55	MULB2	56	MULB3	57	MULW2
58	MULW3	59	MULL2	60	MULLL3
61	EMUL	62	DIVB2	63	DIVB3
64	DIVW2	65	DIVW3	66	DIVL2
67	DIVL3	68	EDIV	69	ASHL
70	ASHQ	71	BITB	72	BITW
73	BITL	74	BISB2	75	BISB3
76	BISW2	77	BISW3	78	BISL2
79	BISL3	80	BICB2	81	BICB3
82	BICW2	83	BICW3	84	BICL2
85	BICL3	86	XORB2	87	XORB3
88	XORW2	89	XORW3	90	XORL2
91	XORL3	92	ROTL		

7.8.2. Die Tests des PVKABB

TEST	BEFEHL	TEST	BEFEHL	TEST	BEFEHL
93	FFC	94	FFS	95	INSV
96	EXTV	97	EXTZV	98	CMPV
99	CMPZV				

7.8.3. Die Tests des PVKABC

TEST	BEFEHL	TEST	BEFEHL	TEST	BEFEHL
100	BNEQ	101	BEQL	102	BGTR
103	BLEQ	104	BGEQ	105	BLSS
106	BGTRU	107	BLEQU	108	BVC
109	BVS	110	BCC	111	BCS
112	BLBC	113	BLBS	114	BBCC
115	BBCS	116	BBSC	117	BBSS
118	BBCCI	119	BBSSI	120	ACBB
121	ACBW	122	ACBL	123	AOBLE
124	AOBLS	125	SOBGE	126	SOBGT

7.8.4. Test des PVKAD

TEST 127 INDEX

7.8.5. Die Tests des PVKABE

TEST	BEFEHL	TEST	BEFEHL	TEST	BEFEHL
128	INSQU	129	INSQH	130	INSQT
131	REMQU	132	REMQH	133	REMQT

7.8.6. Die Tests desPVKABF

TEST	BEFEHL	TEST	BEFEHL	TEST	BEFEHL
134	MOVC3	135	MOVC5	136	MOVTC
137	MOVTU	138	CMPC3	139	CMPC5
140	SPANC	141	SCANC	142	SKPC
143	LOCC	144	MATCH		

7.8.7. Die Tests des PVKABG

TEST	BEFEHL	TEST	BEFEHL	TEST	BEFEHL
145	MOVP	146	CMPP3	147	CMPP4
148	ADDP4	149	ADDP6	150	SUBP4
151	SUBP6	152	MULP	153	DIVP
154	CVTLP	155	CVTPL	156	CVTPT
157	CVTTP	158	CVTPS	159	CVTSP
160	ASHP				

7.8.8. Test des PVKABH

TEST 161 EDITPC

8. PVKAC Gleitkommabefehlstest

8.1. Zusammenfassung

Das Diagnoseprogramm PVKAC, Dok-Nr. 1.57.700004.2, ist ein Programm der Ebene 2. Es ist bestimmt für den Test des internen Gleitkommabefehlsatzes in den verschiedenen Betriebsarten (d.h. die nicht privilegierten Befehle) im RVS-Rechnersystem Robotron K1840.

8.2. Testbedingungen

8.2.1. Hardware

Dieses Programm ist nutzbar auf beliebiger K1840-Rechnerkonfiguration (CPU einschl. Hauptspeicher).

8.2.2. Software

Diagnose-Supervisor

8.3. Voraussetzungen

Mikrodiagnose und Diagnoseprogramm PVKAA (Grundbefehlstest) müssen erfolgreich gelaufen sein.

8.4. Bedienungshinweise

8.4.1. Laden und Starten

Für Laden und Starten siehe Beschreibung "Diagnose-Supervisor". Um die CPU einzubinden, sind folgende Kommandos auszuführen :

für K1840:

```
DS> ATT KAA40 HUB  KA0          (1)(2)(3)(4)

    (1) G_Floating Befehle      (Y oder N)
    (2) H_Floating Befehle      (Y oder N)
    (3) WCS letzte Adresse      (Hex) [ 17FF ohne 2.WCS ]
    (4) Accelerator             (0= FPA nicht vorh.,1= FPA vorhanden)

DS> SEL KA0

DS> R PVKAC
```

8.4.2. Optionen

Die Optionen werden durch das ATTACH-Kommando angewählt. (Accelerator, G_Floating, H_Floating usw.)

ACHTUNG !!

Die Anwahl von G_Floating, H_Floating und Accelerator haben nur Sinn, wenn diese Optionen auch installiert sind !

8.5. Funktionsbeschreibung

8.5.1. Befehlstest-Service-Routine (RITS)

Alle Tests nutzen diese Routine, um die Testdatentabelle zu decodieren, die Testfälle im Speicher aufzubauen, die Befehle auszuführen und die Ergebnisse zu prüfen.
Nach dem Festlegen der Zähler für den jeweiligen Test wird das Dekodierprogramm PVKAC3 aufgerufen, um alle Testdatenpuffer mit den Testdaten aus der Testdatentabelle und der Befehlsdatentabelle zu initialisieren. Danach läuft RITS durch alle Eintragungen in der Adressmodustabelle um die Befehlskette zu bilden, ruft PVKAC4 für jeden Operanden auf und läßt durch PVKAC5 die Befehle ausführen. Eine Exception wird entweder vom Testbefehl oder von der nächsten Anweisung erzwungen (BPT). In diesem Falle wird zu PVKAC6 verzweigt, damit wird der Maschinenstatus gerettet und die Steuerung wieder RITS übergeben.
Am Ende, wenn alle Daten geprüft und ein Fehler erkannt wurde, wird über den Supervisor, der dann seinerseits PVKAC7 aufruft, die normale und die erweiterte Fehlerinformation ausgegeben.

8.5.1.1. RIT Pass 0 Initialisierung

Diese Routine belegt die Datenpuffer für RITS. Außerdem wird der Vektor für den Intervalltimer-Interrupt belegt, falls das Programm im Standalone-Modus läuft.

8.5.1.2. RIT Initialisierungsroutine

Jeder Pass setzt die Pageposition für die Befehlskette und die Operanden auf einen definierten Wert. Am Anfang (Pass 1) ist die Position ein Byte vor der Page-Grenze. Die Befehlskette wird bei jedem Pass um ein Byte nach unten bewegt, der 1. Operand bei jedem 2., der 2.Operand bei jedem 4., ... usw.
Das heißt, um alle Pagepositionen für 6 Operanden zu testen werden 65.576 Durchläufe, bei Einbeziehung der indirekten Operanden 4.194.304 Durchläufe benötigt. Der Stack bewegt sich um 5 Bytes bei jedem Pass, jedoch ist er anfangs nicht in der Nähe einer Page-Grenze eingestellt.

8.5.1.3. RIT Lösch-Routine

Diese Routine löscht alle Datenpuffer, welche bei RITS benötigt werden.

8.5.1.4. RITS

Diese Routine dekodiert die Testdatentabelle mittels einer Befehlsdekodier-Tabelle, verschiedenen Datenspeichern und einer Liste von Speicherpuffern. Sie erzeugt die Befehlskette im Speicher und dann ruft sie die Routine PVKAC5 auf, um den Befehl ausführen zu lassen. Am Schluß werden die Ergebnisse überprüft.

8.5.1.4.1. RIT Parameter-Initialisierung

In diesem Programmteil werden alle Hauptparameter für RIT generiert.

8.5.1.4.2. RIT Testfallauswahl

Hier werden der Testfallzähler und die Schleifensteuerung für die Arbeit mit der ganzen Testtabelle erzeugt.

8.5.1.4.3. RIT Adressmodusauswahl

Mit dieser Sektion werden die Zähler für die Adressmodus-Datentabelle aufgestellt, weiterhin werden hier die Schleifen für alle Entries für jeden Testfall gesteuert.

8.5.1.4.4. Operandendekodierung

Setzt alle aktuellen Datenzähler zurück.

8.5.1.4.5. RIT Wiederanlauf-Routine

Diese Sektion steuert die Anzahl der Wege, auf denen jeder Test wiedergestartet werden kann. Das erste Mal wird sie ohne Faults oder Interrupts, das 2. Mal mit Faults und allen möglichen Speicherreferenzen, das 3. Mal mit Intervalltimer-Interrupts an allen möglichen Punkten und das letzte Mal mit Intervalltimer-Interrupts am Anfang von Page-Faults ausgeführt.
Mit Rücksicht auf die Laufzeit wird die Anzahl der Durchläufe durch das Quick-Flag und die Event-Flags gesteuert.

8.5.1.4.6. RIT Speicherpuffer-Lade-Routine

Setzt alle Speicherpuffer für den Testfall, einschließlich der Hintergrunddaten. Das endgültige Setzen der allgemeinen Register und des Status wird in der PVKAC5-Routine vorgenommen.
Beachte :
Die letzten werden zuerst geladen, so daß die sich überlappenden Operanden mit den richtigen Quelldaten abschließen.

8.5.1.4.7. RIT Testfall-Prüfsektion

Diese Sektion prüft so oft wie möglich den Prozeßstatus nach Beendigung des Befehls bzw. nachdem eine Exception die Abarbeitung gestoppt hat.
Zuerst wird der Maschinenstatus in einen "ACTUAL"-Datenblock gesichert und dann wird jedes Element mit dem korrespondierenden Element aus dem "EXPECTED"-

Datenblock verglichen. Am Schluß werden die Fehler ausgegeben.

8.5.1.5. Testweiterführung

Es existiert die Möglichkeit innerhalb RITS, durch Faults und/oder Interrupts unterbrochene Programme weiterzuführen.
Speicherverwaltungszugriffsverletzungen werden genutzt, um zusätzlich zu den spezifischen Faults und Traps Faults zu erzwingen. Der Intervalltimer erzeugt während der Befehlsabarbeitung nach jeder Mikrosekunde Interrupts bis der Befehl weiter abgearbeitet oder abgeschlossen wird.
Die Testweiterführung kann über die Steuer- und Ereignisflags gesteuert werden. Wenn das QUICK-Flag gesetzt ist, ist die Testweiterführung nicht möglich. Bei gesetztem Ereignisflag 2 ist keine Interrupt-Testweiterführung möglich, aber Speicherverwaltungs-Fault ist möglich.(QUICK-Flag = 0). Es ist dabei zu beachten, daß dieses ständige Interrupts sind, außer wenn diese durch die Diagnosesupervisor-Clock-Routine verboten werden.
Wenn Ereignisflag 3 gesetzt ist (Flag 2 und QUICK = 0), dann kommt der Interrupt zur gleichen Zeit wie der Speicherverwaltungs-Fault.
Unter SVP kann der Intervall-Timer nicht gesteuert werden, d.h. Flag 2 und 3 haben keinen Effekt.

8.5.1.6. Accelerator-Test

Wenn das Programm im standalone-Modus arbeitet, der Accelerator vorhanden ist und durch ATTACH selektiert wurde, dann läuft jeder Test zweimal in jedem Pass. Der Accelerator-Status wird in der Testausschrift angegeben. Der Accelerator wird nicht getestet, wenn er nicht durch ATTACH angegeben ist. In diesem Falle bildet eine Testsequenz einen Pass.

Unter SVP kann der Accelerator nicht durch den Operator beeinflußt werden !!!

8.5.2. Programmgröße

Dieses Programm belegt 59 Kilobytes (118 Pages) im Hauptspeicher.
Der Speicherbereich liegt zwischen 200H und EE00H.

8.5.3. Programmlaufzeit

Der Test benötigt im Standalone-Modus ca. 5:30 min.

8.5.4. Abarbeitungsbedingungen

entfällt

8.5.5. Flag-Verwendung

In diesem Test sind die Ereignisflags 2 bis 6 aktiv.

Flag_2: Verbietet den Intervalltimer-Interrupt während der Abarbeitung des Befehls.

Flag_3: Erlaubt den Intervalltimer-Interrupt, wenn Page-faulting auch erlaubt ist.

Flag_4: Erlaubt die Fortführung eines Subtests nach Fehler (normalerweise wird dieser abgebrochen).

Flag_5: Verbietet Ausführung des Befehles DIVP während Intervalltimer-Interrupt.

Flag_6: Erlaubt dem Nutzer eine eigene Testfolge zusammenzustellen, nachdem angefragt wurde, welche Tests abgearbeitet werden. Wenn dieses Flag gesetzt ist, fordert das Diagnoseprogramm die Nummern der auszuführenden Tests an. Diese werden einzeln eingegeben. Nach der Eingabe wird mit <CR> statt der Testnummer das Programm abgearbeitet. Es kann irgendeine der Testnummern eingeben werden. Wenn eine Teilsektion aufrufen wird und die Testnummer ist nicht in der Sektion, wird der Test nicht ausgeführt (d.h. die Sektion hat höhere Priorität als die Anwahl des Flags 6). Um zu erfahren, welche Befehle in welchem Test ausgeführt werden, kann "HELP PVKAC TESTS" aufgerufen werden.
Dieses Flag arbeitet nicht, wenn das Operator-Flag gelöscht ist!

Anmerkung:
Um die Abarbeitungszeit und den Aufwand zu verringern, werden die Flags nur am Beginn jedes Tests oder nach einem Fehler abgefragt. Deshalb verändert ein Drükken von "^C" und Setzen der Ereignisflags das laufende Programm nicht (außer im Fehlerfalle !).

8.5.6. Allgemeine Programmanforderungen

Das Programm ist lauffähig unter dem Diagnosesupervisor im Standalone-Modus oder als Nutzertask unter SVP.

8.5.7. Fehlermitteilungen

Der Diagnosesupervisor erzeugt Standardausschriften, in denen die Programm-Identifikation, die Testnummer, die Subtestnummer, die Fehlernummer, die Fehlerart, das zu testende Gerät (KA0) und eine von RITS erzeugte Ausschrift, in welcher die fehlerhafte Funktion zu finden ist, enthalten sind.
Die Fehlernummer ist die Adress-Moduseintrittsnummer in der Tabelle (RAM)+1000, d.h. Nr.1005 besagt,daß die 6. Kombination (0 ist die erste !!!) nach Ausführung verlassen wurde, als der Page-Faulting-Interrupt die Ausführung der Anweisungen unterbrochen hat.

Beispiel:

```
************* RVS Floating Point Instr. Test    PVKAC - 1.0  *************
Pass 1,   test 1,   subtest 3,    error 0,    24-DEZ-1986       20:26:06.99

Hard error while testing KA0   Operand data error
```

8.5.7.1. Einfache Fehlerausschrift

Diese Fehlerausschrift enthält den Testbefehl in Macro-Schreibweise einschließlich der Operanden.
Geschah der Fehler während Page-Fault, dann wird dieses angezeigt.
Wenn eine falsche Exception eintrat, wird der erwartete und der aktuelle Wert angegeben. Am Schluß werden alle Operanden und/oder Register, welche fehlerhafte Daten enthalten, ausgegeben. Die Operanden werden als "OPRn" angegeben. Dabei

ist n eine Zahl zwischen 1 und 6 . Falls der Operand eine Adresse bezeichnet, wird diese auf derselben Zeile ausgegeben. Die nächste Zeile enthält dann die Daten (bezeichnet als "@OPR").
Einige Operanden verweisen auf implizite Operanden, diese werden in eckige Klammern gesetzt. Beispiel: "[OPR3]"

Beispiel:

Instruction: ADDD2 (R2),(R3)
with page faults

expected exceptions: Reserved operand data
actual exception : None

	ADDRESS	INITIAL	EXPECTED	ACTUAL	XOR
OPR2	00044DF8				
@OPR2	00044DF8	00000008	00000008	00000009	00000001
OPR4	00044E17	FFFFFFE1	FFFFFFE1	FFFFFFF1	00000010

Registers:

PC	000449FF	000449FF	00044A02	000003FD

8.5.7.2. Erweiterte Fehlerausschrift

Wenn erweiterte Fehlerausschriften erlaubt sind, werden alle wichtigen Informationen über den Testbefehl und von den nicht exakten Daten die fehlerhaften Bytes ausgegeben.
Wenn eine Exception auftrat, so wird dieses angezeigt (außer BPT).
Alle Operanden und alle genutzten Register werden angegeben.

Beispiel:

Instruction: CVTFD (R11),(AP)

exception T-bit Trap

	ADDRESS	INITIAL	EXPECTED	ACTUAL	XOR
OPR1	000557FF	1234C1FF	1234C1FF	1234C1FF	
OPR2	00055BFF	5A	F9	F8	01

Registers:

R11	000557FF	000557FF	000557FF
AP	00055BFF	00055BFF	00055BFF
PC	000553FF	00055402	00055402
PSL	03C00052	03C00058	03C00058

8.5.8. Programmitteilungen

Wenn das Programm mit /QA gestartet wird oder das TRACE-Flag gesetzt ist, werden die jeweils laufenden Tests auf dem Bildschirm aufgelistet. Bei nicht vorhandenem G & H-Gleitkommazusatz (GHE K2813) oder wenn G &H Floating nicht durch den ATTACH-Befehl eingebunden sind, werden die Ausschriften der betreffenden Tests (ab Test 47) modifiziert:

"Test xx xyz Instruction is emulated by SVP, therefore not tested",
wenn K2813 nicht vorhanden ist und

"Test xx xyz Instruction is not ATTACHed, therefore not tested",
wenn G_ bzw. H_Floating nicht eingebunden ist.

8.5.9. Leistungsparameter des Testobjektes

entfällt

8.5.10. Aktionen bei unerwarteten Fehlern

Wenn aus verschiedenen Gründen das Testprogramm Datenfehler in einer der Testtabellen findet oder aus anderen Gründen nicht weiter arbeiten kann, wird durch die Ausschrift "FATAL ERROR" angezeigt, daß ein unvorhersehbarer Fehler das Programm verändert hat und deshalb ein weiterer Test unmöglich ist. Es wird "FATAL ERROR" angezeigt und das Programm abgebrochen. Fehler dieses Typs haben Fehlernummern zwischen 200 und 999, wobei die Hunderter den Programmteil bezeichnen,der den Fehler aufgerufen hat. d.h. Fehler 301 kommt von PVKAC3.

8.5.11. Testfolge bei mehreren Einheiten

entfällt

8.5.12. Einstellungen für Ablauf unter AFT

entfällt

8.5.13. Testsektionen

Den Sektionen sind bestimmte Gruppen von Befehlen zum Testen zugewiesen.

DEFAULT

Alle Tests von Test 1 bis Test 98.

F_FLOATING

Gleitkommabefehle mit einfacher Genauigkeit:

MOVF, MNEGF, TSTF, CVTBF, CVTWF, CVTLF, CVTFB, CVTFW, CVTFL, CVTRFL, CMPF, ADDFn, SUBFn, MULFn, DIVFn, EMODF, POLYF, ACBF

DOUBLE

Gleitkommabefehle mit doppelter Genauigkeit und im erweitertem Bereich doppelter Genauigkeit:

MNEGD, MOVD, TSTD, CVTBD, CVTWD, CVTLD, CVTDB, CVTDW, CVTDL, CVTFD, CVTDF, CVTRDL, CMPD, ADDDn, SUBDn, MULDn, DIVDn, EMODD, POLYD, ACBD, MOVG, MNEGG, CVTGB, CMPG, TSTG, ADDGn, SUBGn, MULGn, DIVGn, EMODG, POLYG, ACBG

D_FLOATING

Gleitkommabefehle mit doppelter Genauigkeit:

MNEGD, MOVD, TSTD, CVTBD, CVTWD, CVTLD, CVTDB, CVTDW, CVTDL, CVTFD, CVTDF, CVTRDL, CMPD, ADDDn, SUBDn, MULDn, DIVDn, EMODD, POLYD, ACBD

EXTENDED

Gleitkommabefehle im erweiterten Bereich doppelter und vierfacher Genauigkeit:

MOVG, MNEGG, CVTGB, CMPG, TSTG, ADDGn, SUBGn, MULGn, DIVGn, EMODG, POLYG, ACBG,
MOVH, MNEGH, CVTHB, CMPH, TSTH, ADDHn, SUBHn, MULHn, DIVHn, EMODH, POLYH, ACBH

G_FLOATING

Gleitkommabefehle im erweiterten Bereich doppelter Genauigkeit:

MOVG, MNEGG, CVTGB, CMPG, TSTG, ADDGn, SUBGn, MULGn, DIVGn, EMODG, POLYG, ACBG

H_FLOATING

Gleitkommabefehle im erweiterten Bereich vierfacher Genauigkeit:

MOVH, MNEGH, CVTHB, CMPH, TSTH, ADDHn, SUBHn, MULHn, DIVHn, EMODH, POLYH, ACBH

MOVXMNEGX

Alle einfachen, doppelten, doppelt erweiterten und vierfach erweiterten MOVx und MNEGx Gleitkommabefehle:

MOVF, MNEGF, MNEGD, MOVD, MOVG, MNEGG, MOVH, MNEGH

TSTX

Alle einfachen, doppelten, doppelt erweiterten und vierfach erweiterten TSTx Gleitkommabefehle:

TSTF, TSTD, TSTG, TSTH

CVTXX

Alle einfachen, doppelten, doppelt erweiterten und vierfach erweiterten CVTxy und CVTxyz Gleitkommabefehle:

CVTBF, CVTWF, CVTLF, CVTFB, CVTFW, CVTFL, CVTRFL, CVTBD, CVTWD, CVTLD, CVTDB, CVTDW, CVTDL, CVTFD, CVTDF, CVTRDL, CVTGB, CVTHB

CMPX

Alle einfachen, doppelten, doppelt erweiterten und vierfach erweiterten CMPx Gleitkommabefehle:

CMPF, CMPD, CMPG, CMPH

ADDSUBMULDIV

Alle einfachen, doppelten, doppelt erweiterten und vierfach erweiterten ADDxn, SUBxn, MULxn und DIVxn Gleitkommabefehle:

ADDFn, SUBFn, MULFn, DIVFn, ADDDn, SUBDn, MULDn, DIVDn, ADDGn, SUBGn, MULGn, DIVGn, ADDHn, SUBHn, MULHn, DIVHn

EMODX

Alle einfachen, doppelten, doppelt erweiterten und vierfach erweiterten EMODx Gleitkommabefehle:

EMODF, EMODD, EMODG, EMODH

POLYX

Alle einfachen, doppelten, doppelt erweiterten und vierfach erweiterten POLYx Gleitkommabefehle:

POLYF, POLYD, POLYG, POLYH

ACBX

Alle einfachen, doppelten, doppelt erweiterten und vierfach erweiterten ACBx Gleitkommabefehle:

ACBF, ACBD, ACBG, ACBH

8.5.14. Verschiedenes

8.5.14.1. Übergehen eines Testfalles

Wenn es aus irgendeinem Grunde nötig ist, einen Test zu umgehen, dann kann die folgende Prozedur durchgeführt werden.

(1) Laden des Programmes
(2) Suchen der Basisadresse der P_SECT des Tests (z.B. TEST002) im Listing dieses Programms.
(3) Setzen des Basisregisters mit diesem Wert.(SET BASE)
(4) Suchen der "TT"- Zeile des richtigen Testfalles und Überprüfung, ob die richtige Stelle gefunden wurde.
(5) Zählen der Anzahl der Bytes in der Expansion des Testfalles und Subtrahieren von 3.
(6) Schreiben der Adresse aus Schritt 4 + dem Wert aus Schritt 5 in ein Langwort ab 03FC (DEPOSIT LONGWORD).
(7) Rücksetzen des Relocation-Registers zurück auf 0.(SET BASE 0)

8.5.14.2. First Part Done (FPD)

8.5.14.2.1. Was ist FPD ?

FPD ist der Name des 27. Bits im PSL. Es liefert die Fähigkeit, lange andauernde Befehle sicher zu unterbrechen und diese später fortzuführen. Nur wenige Befehle sind in diesem Sinne zu unterbrechen.
Die meisten Befehle erkennen den Interrupt vor ihrer Abarbeitung oder mitten in der Operation, worauf der Befehl abgebrochen und der Zustand vor Befehlsbeginn wiederhergestellt wird; d.h. es wird der Zustand hergestellt, als wenn der Interrupt vor Befehlsbeginn erkannt wurde. Einige Befehle sind aber so lang, daß dieses nicht ausführbar ist. Dieses sind die Zeichen- und Dezimalkettenoperationen POLYF, POLYD und CRC. Jede dieser Operationen schreibt einen Kontrollblock in die Hauptregister. Wenn ein Interrupt während der Abarbeitung angemeldet wird, wird der aktuelle Befehlsstatus (d.h. was gerade getan wurde und wie weit der Befehl bereits abgearbeitet ist) in diesen Kontrollblock gerettet um nach Interruptabarbeitung diesen wiederzuerhalten. Der Befehl setzt dann FPD im PSL, damit danach der Befehl weiter abgearbeitet werden kann. In diesem Falle holt sich der Befehl den Status wieder von den Hauptregistern und beginnt an dem Punkt, wo er unterbrochen wurde.

8.5.14.2.2. Wie kann es zu Fehlern kommen ?

Das Mikroprogramm, das die FPD-Möglichkeit ausführt, muß in der Lage sein, den Status korrekt abzuspeichern und wiederzuholen, egal welcher Speicherbereich gerade bearbeitet wird und woher der Interrupt kommt. Der Status der Operationen ist in diesem Falle kompliziert und es kann vorkommen, daß das Mikroprogramm nicht alles korrekt abspeichert oder zurückholt. Wenn die Operation später fortgesetzt wird, kann es möglich sein, daß der Maschinenstatus (in Form von Internregistern) während der Unterprogrammabarbeitung verändert wurde und damit nicht mehr korrekt ist. Dies führt zu unvorhersehbaren Fehlern, meistens in Form von falsch geschriebenen Daten, falschen Zeichenkettenlängen und Bedingungscodes etc., welche schnell erkannt werden können. Falls während der Interruptabarbeitung die Internregister nicht verändert wurden, wird der Fehler nicht erkannt. Um dieses zu vermeiden, wird während der Interruptbehandlung eine Routine abgearbeitet, welche die internen Register modifiziert. Diese Routine ist stark

Abarbeitung abhängig.
Der Befehl kann sowohl beim Retten des Status als auch beim Setzen des FPD zu Fehlern führen. Das passiert normalerweise dann, wenn modifizierte Register als Argumente beim Restart der Anweisung benutzt werden. Dieses wird in der Interruptroutine erkannt, um damit abzusichern, daß wenn FPD gelöscht ist, die Originalargumente nicht verändert wurden.

8.5.14.2.3. Was wird getan, um dies zu testen ?

Jeder dieser Befehle testet während seiner Abarbeitung, ob Interrupts angefordert werden. Um den Mikroprogrammtest zu kontrollieren und die Interrupts zu behandeln, benötigt jeder Befehl die vom Intervalltimer erzeugten Interrupts. Der Befehl wird im Endeffekt vollständig abgearbeitet, aber er muß regelmäßig mit diesen Interrupts unterbrochen werden.
Die Anweisungen können sowohl mit Interrupts als auch durch Page-Faults getaktet werden. Ein Befehl hat bis zu 6 Operanden, somit werden 12 Pages mit diesen belegt. Damit ist ermöglicht, daß jeder Operand einen unabhängigen Bereich zugeordnet bekommt. Wenn ein Befehl begonnen wird, ist jede Page ungültig. Wenn versucht wird, auf die Operanden zuzugreifen, wird ein Fault erzeugt. Jeder Fault macht die zugehörige Page gültig, so daß die Befehlsabarbeitung voranschreitet. Dies allein sichert aber noch nicht, daß diese Page vollständig getestet ist. Sobald auf eine Page das erste Mal zugegriffen wird, so werden alle anderen Pages ungültig. Dieses wird in jedem Falle überprüft und wenn jede Page den ersten Zugriff einmal hatte, wird der Befehl fertig ausgeführt.
Als Beispiel wird ein Befehl mit 2 String-Operanden und einem Nicht-string-Operanden betrachtet.
Bei diesem Befehl soll der Nicht-String-Operand und dann die String-Operanden verarbeitet werden (z.B. CMPC3).
Zuerst wird auf den Nicht-String-Operanden zugegriffen und die Page, die ihn enthält, mit einem Fault belegt. Nach Restart wird er vom Befehl problemlos geholt und die String-Verarbeitung beginnt. Da jeder Operand an einer Page-Grenze beginnt, tritt bei jedem Zugriff ein Fault auf. Der Befehl greift erst auf die erste Page des ersten Operanden, dann auf die erste Page des zweiten Operanden, dann auf die 2. Page des 1. Operanden und dann auf die 2. Page des 2.Operanden zu. Weil beim 1. Zugriff auf eine Page ein Fault auftritt, erkennt der Test, daß er alle anderen Pages ungültig machen muß.
Die folgende Tabelle veranschaulicht dieses.

page 1	page 2	page 3	page 4	
u	u	u	u	Alle Pages sind am Anfang ungültig.
fault	u	u	u	Die erste Page hat Fault
g	u	u	u	und ist gültig.
g	u	fault	u	Die erste Page von Operand 2
u	u	g	u	wird gültig, der Rest ung.
fault	u	g	u	2. Zugriff auf Page 1,
g	u	g	u	Page 3 bleibt gültig
g	fault	g	u	Arbeitsbereich String 1 ist
u	g	u	u	Page 2, Rest ungültig.
u	g	fault	u	Page 1 wird jetzt nicht
u	g	g	u	benötigt, aber Page 3.
u	g	g	fault	Bereich String 2, seine 2.
u	u	u	g	Page macht Page 4 gültig.
u	fault	u	g	Page 2 wird benötigt und
u	g	u	g	wieder gültig gemacht.
u	g	u	g	Der Befehl wird fertig abgearbeitet.

8.6. Testbeschreibungen

Im Gleitkommabefehlstest existieren folgende Programmodule:

PVKAC1 - Header Modul
PVKAC2 - RITS (Instruction Test Services)
PVKAC3 - RIT_DECODE
PVKAC4 - RIT_OPERANDS
PVKAC5 - RIT_EXECUTE
PVKAC6 - RIT_EXCEPTION
PVKAC7 - ERROR REPORTING ROUTINE
PVKAC8 - TEST INSTRUCTION DECODE DISPATCH TABLES
PVKAC9 - GLOBAL TEST DATA POOLS
PVKACA - Gleitkommabefehle einfacher Genauigkeit (F_Floating)
PVKACB - Gleitkommabefehle doppelter Genauigkeit (D_Floating)
PVKACC - Gleitkommabefehle im erweiterten Bereich doppelter Genauigkeit (G_Floating)
PVKACD - Gleitkommabefehle im erweiterten Bereich vierfacher Genauigkeit (H_Floating)

Jedes der oben angegebenen Module enthält sämtliche Informationen über die jedem Modul zugeordneten Befehle, die zur Dekodierung eines Testtabelleneintrages und zur Ausführung der Befehle benötigt werden. Die zu dekodierende Dispatchliste wird von PVKAC3 genutzt. Das erste Byte der Liste gibt die Länge der Liste und das zweite Byte die Länge des Befehls in Byte an. Die restlichen Bytes sind "Hauptstatus"-Indexe die zur Steuerung von PVKAC3 bei dessen Interpretation der Testtabelleneinträge genutzt werden. Diese Indexe werden durch im Speicher stehende Operanden aufgerufen und sie schreiben den Befehl in den Speicher. Weiterhin kellern sie die Rücksprungadresse ein, geben die Befehlsadresse im Stack an und übernehmen die Ausführung eines REI. (Beim PVKACA folgt der zu dekodierenden Dispatchliste eine ASCIC-Zeichenkette welche die Befehlsmnemonik enthält. Diese wird bei der Fehlerreporting-Routine genutzt.)
Der Zugriff zum Feld, welches die obigen Informationen enthält, wird mittels der Tabelle "RID OFFS.TAB", welche Teil des RITS- Moduls ist, realisiert. Die Tabelle wird von einem Operationscode indexiert und dessen Wortinhalt wird zur Adresse "RID AREA" addiert. Das geschieht, um die Adresse der Dekodier-Dispatchliste für einen Testbefehl zu bilden.
Die jeweiligen einzelnen 98 Tests überprüfen die Operationen des Microcodes für die zu testenden Befehle, die Befehlszwischenspeicherung und die Bedingungscodelogik. Die beiden letztgenannten Funktionen sind bei Befehlen gleicher Art gleich.

8.7. Abkürzungsverzeichnis

BPT	Break-Point-Trap
FPA	Gleitkommabeschleuniger
FPD	First Part Done (Bit 27 im PSL)
GHE	G & H Gleitkommazusatz K2813
PSL	Prozessorstatus-Langwort
PVKAA	Grundbefehlstest
PVKAB	Architektur-Befehlstest
PVKAC	Gleitkommabefehlstest
SVP	Virtuelles Betriebssystem der RVS-K1800 Systeme
WCS	ladbarer Mikroprogrammspeicher
RAM	Adressmodus
RIT	Befehlstest

8.8. Auflistung der Tests

Das Programm umfaßt 98 Befehlstests, diese sind in 4 Gruppen gegliedert.

Test 1 - 22 Befehle einfacher Genauigkeit (F_Floating)

Test 1	MOVF	Test 9	CVTFL	Test 17	MULF3
Test 2	MNEGF	Test 10	CVTRFL	Test 18	DIVF2
Test 3	TSTF	Test 11	CMPF	Test 19	DIVF3
Test 4	CVTBF	Test 12	ADDF2	Test 20	EMODF
Test 5	CVTWF	Test 13	ADDF3	Test 21	POLYF
Test 6	CVTLF	Test 14	SUBF2	Test 22	ACBF
Test 7	CVTFB	Test 15	SUBF3		
Test 8	CVTFW	Test 16	MULF2		

Test 23 - 46 Befehle doppelter Genauigkeit (D_Floating)

Test 23	MNEGD	Test 31	CVTDL	Test 39	SUBD3
Test 24	MOVD	Test 32	CVTFD	Test 40	MULD2
Test 25	TSTD	Test 33	CVTDF	Test 41	MULD3
Test 26	CVTBD	Test 34	CVTRDL	Test 42	DIVD2
Test 27	CVTWD	Test 35	CMPD	Test 43	DIVD3
Test 28	CVTLD	Test 36	ADDD2	Test 44	EMOD
Test 29	CVTDB	Test 37	ADDD3	Test 45	POLYD
Test 30	CVTDW	Test 38	SUBD2	Test 46	ACBD

Test 47 - 70 Befehle im erweiterten Bereich doppelter Genauigkeit (G_Floating)

Test 47	MOVG	Test 55	CVTLG	Test 63	SUBG3
Test 48	MNEGG	Test 56	CVTFG	Test 64	MULG2
Test 49	CVTGB	Test 57	CVTGF	Test 65	MULG3
Test 50	CVTGW	Test 58	CMPG	Test 66	DIVG2
Test 51	CVTGL	Test 59	TSTG	Test 67	DIVG3
Test 52	CVTRGL	Test 60	ADDG2	Test 68	EMODG
Test 53	CVTBG	Test 61	ADDG3	Test 69	POLYG
Test 54	CVTWG	Test 62	SUBG2	Test 70	ACBG

Test 71 - 98 Befehle im erweiterten Bereich vierfacher Genauigkeit (H_Floating)

Test 71	MOVH	Test 81	CVTDH	Test 91	SUBH3
Test 72	MNEGH	Test 82	CVTGH	Test 92	MULH2
Test 73	CVTHB	Test 83	CVTHF	Test 93	MULH3
Test 74	CVTHW	Test 84	CVTHD	Test 94	DIVH2
Test 75	CVTHL	Test 85	CVTHG	Test 95	DIVH3
Test 76	CVTRHL	Test 86	CMPH	Test 96	EMODH
Test 77	CVTBH	Test 87	TSTH	Test 97	POLYH
Test 78	CVTWH	Test 88	ADDH2	Test 98	ACBH
Test 79	CVTLH	Test 89	ADDH3		
Test 80	CVTFH	Test 90	SUBH2		

9 PVKAD Kompatibilitätsmodus-Befehlstest

9.1. Zusammenfassung

Das Diagnoseprogramm PVKAD (Dok.Nr.: 1.57.700005.0) ist ein Programm der Ebene 2 (Stand-alone-Modus und User-Modus). Es ist bestimmt für die Testung des gesamten Befehlssatzes des Kompatibilitätsmodus des RVS Rechnersystems K 1840. Gegenüber dem SKR-Befehlssatz gelten folgende Einschränkungen.
Nicht möglich ist:

1. Die Abarbeitung von privilegierten Befehlen, wie HALT und RESET, sowie speziellen Befehlen, wie TRAPS und WAIT und der Gleitkommabefehle.
2. Zugriff zu internen Prozessorregistern, wie PSW.
3. Direktzugriff zu TRAP- und Interruptvektoren.
4. Direktzugriff zu E/A-Geräten.
5. Interruptarbeit.

Die Programmabarbeitung ist nur unter Steuerung des Diagnose-Supervisors möglich:

1. Stand-alone-Modus.
2. User-Modus (Nutzerprogramm unter SVP).

9.2. Testbedingungen

9.2.1. Hardware

K 1840-Rechnerkonfiguration aus CPU K 2810 (KAA40) und Hauptspeicher K 3581.10 (MSA40), arbeitsfähig in der Ebene 2.

9.2.2. Software

Diagnose-Supervisor arbeitsfähig im Stand-alone-Modus

9.3. Voraussetzungen

Der Grundbefehlstest PVKAA und der Grundspeichertest PVKAS müssen abgearbeitet worden sein. Dies erfolgt in der Ebene 4. Zuvor muß jedoch auch die erfolgreiche Abarbeitung der Mikrodiagnose in der Ebene 5 stattgefunden haben.

9.4. Bedienungshinweise

Man beziehe sich auf die Bedienungsunterlagen für den Diagnose-Supervisor im Diagnosehandbuch, um den Diagnose-Supervisor zu laden und zu starten, sowie das Testprogramm PVKAD laden und unter der Steuerung des Diagnose-Supervisors abarbeiten zu können. Vor dem Start ist der vorhandene CPU-Typ einzubinden und für den Test logisch auszuwählen. Dies erfolgt mit den Kommandos ATTACH (einbinden) und SELECT (auswählen). Es werden im Testablauf zwei Konfigurationen unterschieden. Konfigurationen mit und ohne Gleitkommabeschleuniger (FPA). Die Einbeziehung des FPA in den Test erfolgt ausschließlich im Stand-alone-Modus bei nicht gesetzten Ereignisflag 1 im FPA-Steuer-/Status-Register ACCS. Dieser Teil des Tests wird unterbunden, wenn vor dem Start das Flag 1 gesetzt wird. Wenn das Programm als Nutzerprogramm unter der Regie des Betriebssystems SVP läuft, erfolgt ebenfalls keine FPA-Prüfung.

9.5. Funktionsbeschreibung

9.5.1. Übersicht

Das Programm besteht aus 4 Tests, in denen die verschiedenen Gruppen von Kompatibilitätsmodus-Befehlen bzw. Funktionen getestet werden.

Test 1: Test des Übergangs zum Kompatibilitätsmodus bzw. Beendigung desselben.
Dieser Test prüft die Exceptions, die die Umschaltung zwischen den zwei Arbeitsmodi bewirken. Es werden alle Kompatibilitätsmodus-Exceptions geprüft.

Test 2: Test der Einoperandenbefehle.
Dieser Test ist in zwei Teile aufgeteilt. Im Teil 1 ist dem Speicher, auf den die zu testenden Befehle zugreifen, der Status "Schreiben" gegeben und die Befehle werden auf ihre eigene Abarbeitung hin geprüft. Im Teil 2 ist dem Speicher, auf den die zu testenden Befehle zugreifen, der Status "Lesen" gegeben. Alle Einoperandenbefehle werden erneut durchgeführt, um die zu erwarteten Zugriffsverletzungen überprüfen zu können.
Im Test sind folgende Befehle enthalten:

CLR(B)	Löschen
DEC(B)	Dekrement
INC(B)	Inkrement
NEG(B)	Negation
TST(B)	Test
COM(B)	Komplement
ASR(B)	Arithmetische Rechtsverschiebung
ASL(B)	Arithmetische Linksverschiebung
ADC(B)	Addiere C-Bit
SBC(B)	Subtrahiere C-Bit
ROR(B)	Logische Rechtsverschiebung
ROL(B)	Logische Linksverschiebung
SXT	Vorzeichenerweiterung
SWAB	Bytevertauschung

MFPI	Transport aus vorhergehendem Befehlsbereich
MFPD	Transport aus vorhergehendem Datenbereich
MTPI	Transport in vorhergehenden Befehlsbereich
MTPD	Transport in vorhergehenden Datenbereich

Test 3: Test der Zweioperandenbefehle.
Dieser Test testet alle Zweioperandenbefehle entsprechend Teil 1 des Tests 2.
Im Test sind folgende Befehle enthalten:

MOV(B)	Transport
CMP(B)	Vergleich
BIS(B)	Bit setzen
BIT(B)	Bit testen
BIC(B)	Bit löschen
ADD	Addieren
SUB	Subtrahieren
XOR	Exclusiv-Oder
MUL	Multiplizieren
DIV	Dividieren
ASH	Arithmetische Verschiebung
ASHC	Arithmetische Doppelwortverschiebung

Test 4: Test der Programmsteuerbefehle.
Dieser Test testet alle im Kompatibilitätsmodus ausführbaren Programmsteuerbefehle.
Im Test sind folgende Befehle enthalten:

JMP	Sprung unbedingt
JSR	Unterprogrammsprung
RTS	Rücksprung vom Unterprogramm
SOB	Zyklusbefehl

alle Flagbit-Behandlungsbefehle
alle Branch-Befehle

9.5.2. Programmgröße

Das Programm benötigt 123 Pages (Speicherseiten), genau 62.976 Kbyte.

9.5.3. Programmlaufzeit

Die Laufzeit bei fehlerfreiem Durchlauf im Stand-alone-Modus beträgt ca. 25 Sekunden.

9.5.4. Abarbeitungsbedingungen

entfällt

9.5.5. Flag-Verwendung

Flag 1 unterscheidet den Testablauf mit und ohne Einbeziehung des FPA. Vor Abarbeitung des Programms im Stand-alone-Modus wird das FPA-Steuer-/Status-Register ACCS geprüft. Ist ein FPA (Option) vorhanden, so läuft die vollständige Testfolge in zwei Durchläufen, einmal mit eingeschaltetem und einmal mit ausgeschaltetem FPA. Im 1.Durchlauf wird der FPA-Status in einer Ausschrift angegeben. Diese Einbeziehung des FPA in den Test wird bei gesetzten Flag 1 unterbunden. Es erfolgt dann nur 1 Durchlauf der Testfolge.

9.5.6. Allgemeine Programmanforderungen

entfällt

9.5.7. Fehlerermittlung

Der Diagnose-Supervisor erzeugt im Fehlerfall eine Standard-Fehlerausschrift, die die Programmidentifikation (ID), die Routinenummer, die Subroutinenummer, die Fehlernummer, die Fehlerklasse (HARD), die im Test befindliche Funktionseinheit (KAO) und darüberhinaus eine Mitteilung enthält, die den gerade getesteten Befehl ausweist.

```
*** RVS Compatability Mode Instructions Exerciser PVKAD 1.0 ***
PASS 1  TEST 2  SUBTEST 3  ERROR 1    08-APR-1987
HARD ERROR WHILE TESTING KAO :(Angaben über den zu testenden Befehl)
```

Bei Auftreten eines Fehlers erfolgt gegebenfalls im Stand-alone-Modus die Ausgabe des FPA-Status, generell aber erfolgt die Ausgabe des zu testenden Befehls mit seinen Operanden.

Beispiel:

```
INSTRUCTION EXECUTED : MOVB @0(R1), @2(R3)
```

In Ergänzung hierzu zeigt eine erweiterte Fehlerausschrift alle Initialwerte sowie die Soll- und Istwerte aller Register, des PSL und aller einbezogenen Quell- und Zieloperanden an.

Beispiel:

	R0	R1	R2	R3	R4	R5	SP	PSL	DST	SRC
INITIAL :	0000	AC2C	2222	AC20	4444	5555	0200	83C0000E	FFFF	FF00
EXPECTED:	0000	AC2C	2222	AC20	4444	5555	0200	83C00004	FF00	FF00
ACTUAL :	0000	AC2C	2222	AC20	4444	5555	0200	83C00008	FFF4	FF00

Bemerkung: Die Spalten Quell- und Zieldaten werden gedruckt, sofern sie im Befehl enthalten sind. Die Angaben erfolgen hexadezimal, also mit 4 Ziffern, das PSL benötigt 8 Ziffern.

Der aufgetretene Fehler ist in den meisten Fällen mittels der angegebenen Test-, Subtest- und Fehlernummer bzw. der zusätzlichen Angaben über den zu testenden Befehl und unter Zuhilfenahme des Programmassemblerausdruckes eindeutig zu analysieren.
Um im Speicher Daten vom SKR-Code prüfen oder eingeben zu können, muß in folgenden Schritten vorgegangen werden:

1. Aufsuchen der Anfangsadresse der Programmsektion (PSECT) des als fehlerhaft ausgewiesenen Tests aus der Assemblerliste und Setzen des Basisregisters auf diese Adresse. Die Programmsektionen dieses Programms sind TEST 01...TEST 04.
2. Setzen Ausgabe in oktale Form.
3. Prüfen oder Eingabe von Daten in oktaler Form.

Beispiel:

Für PSECT TEST 02 findet man in der Assemblerliste die Adresse 224C. also:

```
DS> SET BASE 224C
DS> SET DEF  OCTAL
DS> SET DEF  WORD
```

Nun sucht man die oktale Adresse in der Assemblerliste, prüft und speichert in oktaler Form.

Bemerkung:

Unterbrechungspunkte innerhalb des SKR-Code sind im Diagnose-Supervisor nicht enthalten.

9.5.8. Programmitteilungen

entfällt

9.5.9. Leistungsparameter des Testobjekts

entfällt

9.5.10. Aktionen bei unerwarteten Fehlern

entfällt

9.5.11. Testfolge bei mehreren Einheiten

entfällt

9.5.12. Einstellungen für Ablauf unter AFT

entfällt

9.5.13. Testsektionen

Das Programm unterteilt sich in drei Testsektionen, die durch Eingabe der betreffenden Sektionsnamen aufrufbar sind.

1. DEFAULT : Standard-Gesamttest
2. EXCEPTIONS : Teiltest
3. INSTRUCTIONS : Teiltest-Befehle

9.5.13.1. Tests und Subtests

Das Programm enthält folgende Tests, Subtests und Routinen.

PVKAD0: Zentrale Module

Routinen:
- Zentraler Programmdatenblock
- Dispatch-Tabelle
- Hardware-Programmtabelle
- Globale Programmdaten
- Initialisieren
- Systemblock zurücksetzen
- Zusammenfassender Bericht

PVKAD1: Teststeuerung Test 1

Test 1: Ein- und Ausschalten des Kompatibilitätsmodus
Test 1.1.: Exceptions durch TRAP-Befehle
Test 1.2.: Exception durch T-Bit-TRAP
Test 1.3.: Exceptions durch RTT und RTI
Test 1.4.: Exception durch ungerade Adresse
Test 1.5.: Exception durch fehlerhafte Befehlskodierung

Routinen:
- GEN/HNDLR: Ablegen von Daten im Stack, Auslösen einer Exception im Kompatibilitätsmodus.
- ERR RPT : Aufstellung und Ausgabe von Spaltenüberschriften und Fehlermitteilungen.

PVKAD2: SKR-Testroutinen für Test 1 (Exceptions siehe PVKAD1)

PVKAD3: Teststeuerung Test 2, 3 und 4

Test 2: Einoperandenbefehle
Test 3: Zweioperandenbefehle
Test 4: Steuerbefehle

PVKAD4: Bedienungsroutinen für den Kompatibilitätsmodus-Befehlstest Test 1, 2, 3 und 4

Routinen:
- Exception-Behandlung für RESET, BPT, IOT, EMT, TRAP (Code-Entschlüsselung u.a.)
- RESET
- BPT
- IOT
- EMT
- TRAP
- Adresse oder Codierung fehlerhaft
- Daten-Vorgabe für JMP und BRANCH
- Daten-Vorgabe für Einoperandenbefehle
- Daten-Vorgabe für Zweioperandenbefehle
- Daten-Vorgabe für Doppelwortbefehle
- Daten-Vorgabe für MTPI und MFPI
- Dateneintrag in Speicher und Register
- Datenprüfung
- Soll-Datenvorgabe
- Initialisieren aller Register
- Fehlerausgabe

PVKAD5: SKR-Einoperandenbefehle

Test 2: Einoperandenbefehle

Subtest:1	CLR R	Registermodus, Modus 0
2	CLR (R)	Register indirekt, Modus 1
4	CLR @(R)+	Autoinkrement indirekt, Modus 3
5	CLR -(R)	Autodekrement, Modus 4
6	CLR @-(R)	Autodekrement indirekt, Modus 5
7	CLR X(R)	Indexierung, Modus 6
8	CLR @X(R)	Indexierung indirekt, Modus 7
9	CLR #N	Direktoperand, Modus 2, Register 7
10	CLR @#A	Absolute Adresse, Modus 3, Register 7
11	CLR A	Relative Adresse, Modus 6, Register 7
12	CLR @A	Relative Adresse, Modus 7, Register 7
13- 24	CLRB	analog Subtest 1-12
25- 48	DEC(B)	analog Subtest 1-24
49- 72	INC(B)	analog Subtest 1-24
73- 96	NEG(B)	analog Subtest 1-24
97-120	TST(B)	analog Subtest 1-24
121-144	COM(B)	analog Subtest 1-24
145-168	ASR(B)	analog Subtest 1-24
169-192	ASL(B)	analog Subtest 1-24
193-216	ADC(B)	analog Subtest 1-24
217-240	SBC(B)	analog Subtest 1-24
241-264	ROL(B)	analog Subtest 1-24
265-288	ROR(B)	analog Subtest 1-24
289-300	SXT	analog Subtest 1-12
301-312	SWAB	analog Subtest 1-12
313-318	MTPI	analog Subtest 1- 6
319-324	MTPD	analog Subtest 1- 6
325-330	MFPI	analog Subtest 1- 6
331-336	MFPD	analog Subtest 1- 6
337-342	MTPI	analog Subtest 7-12
343-348	MTPD	analog Subtest 7-12
349-354	MFPI	analog Subtest 7-12
355-360	MFPD	analog Subtest 7-12

PVKAD6: SKR-Doppeloperandenbefehle

Test 3: Doppeloperandenbefehle

Subtest:1	MOV #N,R	Modus 2, Register 7; Modus 0
2	MOV R,R	Modus 0; Modus 0
3	MOV (R),(R)	Modus 1; Modus 1
4	MOV (R)+,(R)+	Modus 2; Modus 2
5	MOV @(R)+,@(R)+	Modus 3; Modus 3
6	MOV -(R),-(R)	Modus 4; Modus 4
7	MOV @-(R),@-(R)	Modus 5; Modus 5
8	MOV X(R),X(R)	Modus 6; Modus 6
9	MOV @X(R),@X(R)	Modus 7; Modus 7
10	MOV @#A,@#A	Modus 3, Register 7; Modus 3, Register 7
11	MOV A,A	Modus 6, Register 7; Modus 6, Register 7
12	MOV @A,@A	Modus 7, Register 7; Modus 7, Register 7
13	MOV (R),#N	Modus 1; Modus 2, Register 7
14- 25	MOVB	analog Subtest 1,3-9,13,10-12
26- 37	CMP	analog Subtest 2-9,1,10-12

38	CMP R,#N	Modus 0; Modus 2, Register 7
39	CMPB #N,#N	Modus 2, Register 7; Modus 2, Register 7
40- 50	CMPB	analog Subtest 2-12
51- 62	BIT	analog Subtest 1,38,3-12
63- 74	BITB	analog Subtest 1,38,3-12
75- 86	BIC	analog Subtest 1,38,3-12
87- 98	BICB	analog Subtest 1,38,3-12
99-110	BIS	analog Subtest 1,38,3-12
111-122	BISB	analog Subtest 1,38,3-12
123-134	ADD	analog Subtest 1,38,3-12
135-146	SUB	analog Subtest 1,38,3-12
147	XOR R,R	Modus 0; Modus 0
148	XOR R,(R)	Modus 0; Modus 1
149	XOR R,(R)+	Modus 0; Modus 2
150	XOR R,@(R)+	Modus 0; Modus 3
151	XOR R,-(R)	Modus 0; Modus 4
152	XOR R,@-(R)	Modus 0; Modus 5
153	XOR R,X(R)	Modus 0; Modus 6
154	XOR R,@X(R)	Modus 0; Modus 7
155	XOR R,#N	Modus 0; Modus 2, Register 7
156	XOR R,@#A	Modus 0; Modus 3, Register 7
157	XOR R,A	Modus 0; Modus 6, Register 7
159	ASH R,R	Modus 0; Modus 0
160	ASH (R),R	Modus 1; Modus 0
161	ASH (R)+,R	Modus 2; Modus 0
162	ASH @(R)+,R	Modus 3; Modus 0
163	ASH -(R),R	Modus 4; Modus 0
164	ASH @-(R),R	Modus 5; Modus 0
165	ASH X(R),R	Modus 6; Modus 0
166	ASH @X(R),R	Modus 7; Modus 0
167	ASH #N,R	Modus 2, Register 7; Modus 0
168	ASH @#A,R	Modus 3, Register 7; Modus 0
169	ASH A,R	Modus 6, Register 7; Modus 0
170	ASH @A,R	Modus 7, Register 7; Modus 0
171-182	ASHC	analog Subtest 159-170
183-194	MUL	analog Subtest 159-170
195-206	DIV	analog Subtest 159-170

PVKAD7: SKR-Steuerbefehle
Test 4: SKR-Steuerbefehle

Subtest:1 BR mit positivem und negativem Offset
2 bedingter Branch, Prinziptest für Flags
3 BNE für Z= 1,0
4 BEQ für Z= 1,0
5 BPL für N= 1,0
6 BMI für N= 1,0
7 BVC für V= 1,0
8 BVS für V= 1,0
9 BCC für C= 1,0
10 BCS für C= 1,0
11 BGE für N,V= 00, 01, 10, 11
12 BLT für N,V= 00, 01, 10, 11
13 BGT für Z= 1 und N,V= 00, 01, 10
14 BGT für Z= 0 und N,V= 00, 01, 10
15 BLE für Z= 1 und N,V= 00, 01, 10
16 BLE für Z= 0 und N,V= 00, 01, 10
17 BHI für Z,C= 00, 01, 10, 11

```
    18 BHIS für C= 0,1
    19 BLOS für Z,C= 00, 01, 10, 11
    20 BLO für C= 0,1
21- 31 JMP analog Test 2 Subtest 2-12
32- 42 JSR analog Test 2 Subtest 10, 2-9, 11-12
    43 RTS
    44 SOB
    45 Flags "Setzen" und "Löschen"
    46 NOP
```

9.5.14. Verschiedenes

entfällt

9.6. Testbeschreibungen

9.6.1. PVKAD0: Zentrale Module

Diese Module stellen alle Standardinformationen und Routinen bereit, so wie sie vom Diagnose-Supervisor angefordert werden.

9.6.1.1. Zentraler Programmdatenblock

Dieser Datenblock enthält Parameter, die dem Diagnose-Supervisor zur Programmsteuerung dienen. Der Diagnose-Supervisor holt diese Informationen vom Speicher, auf der Adresse 200 (hex) beginnend.

9.6.1.2. Dispatch-Tabelle

Diese Tabelle enthält Adressen, die zu Beginn eines jeden Tests generiert und durch den Linker zusammengefaßt werden. Dies erfolgt durch die Programmsektion DISPATCH.

9.6.1.3. Hardware-Parametertabelle

Diese Tabelle enthält alle hardwarebezogenen Parameter, die während der Programmlaufzeit modifiziert werden können. Der Diagnose-Supervisor stellt für jeden Test eine seperate Kopie spezifischer Parameter her.

9.6.1.4. Globale Programmdaten

Diese Sektion enthält allgemeine Daten, die in mehr als einem Modul verwendet werden.

9.6.1.5. Programmausschriften

Diese Sektion enthält die von allen Tests genutzten Zeichenkettendefinitionen.

9.6.1.6. Initialisierung

Diese Routine wird zu Beginn jeder Wiederholung eines Tests ausgeführt. Sie hat bei Arbeitsweise im Stand-alone-Modus und bei Vorhandensein eines FPA die Aufgabe, den doppelten Durchlauf der Tests zu steuern. Der Diagnose-Supervisor fordert diese Routine automatisch mit einer CALL-Instruktion an.

9.6.1.7. Systemkontrollblock rücksetzen

Diese Routine dient unter anderen zur Freigabe von Vektoren.

9.6.1.8. Zusammenfassender Bericht

Diese Routine zeigt das Testergebnis bereits abgearbeiteter Teiltests an.

9.6.2. PVKAD1: Kompatibilitätsmodus-Exceptions

Diese Routine testet die verschiedenen Kompatibilitätsmodus-Exceptions, insbesondere prüft sie den Mechanismus der RVS-Architektur, den SKR-Kompatibilitätsmodus ein- und auszuschalten.

9.6.2.1. Test 1 Kompatibilitätsmodus, Eintritt und Austritt

Diese Routine läuft in Vorbereitung der fünf Subtests, die nachfolgend aufgeführt werden. Sie prüft den Arbeitsmodus der CPU, stellt gegebenfalls den User-Modus ein und verzweigt zu allen fünf Subtests. Die Einstellung des User-Modus erfolgt durch Rücksetzen des Prozessorstatuslangwortes PSL (Bit 31).

9.6.2.2. Subtest 1.1. Exception durch TRAP-Befehle

Dieser Subtest prüft bei Eintritt in den Kompatibilitätsmodus folgende TRAP-Befehle und reservierte Befehlskodierungen:

BPT
IOT
EMT
HALT
WAIT
RESET
SPL
MARK
FADD
FSUB
FMUL
FDIV
7000-7777 (reservierte Codes)
107000-107777 (reservierte Codes)

Der Test erfolgt in folgenden Schritten:

1. Initialisieren von R8 mit der Anfangsadresse der Tabelle GEN-BLK, die für die Übergabe von Daten in den Stack zuständig ist. .
2. Initialisieren der Tabelle GEN-BLK mit den notwendigen Daten für den unter Test stehenden TRAP-Befehl (siehe Routine GEN im Abschnitt 9.6.2.6.).
3. Aufruf der Routine GEN.
4. Fehlerprüfung und gegebenfalls Protokollierung.
5. Endeprüfung und Verzweigung zum nächsten Subtest, sofern der laufende Subtest abgearbeitet ist, andernfalls Verzweigung zum Testschritt "2." für die Abarbeitung der nächsten Testfunktion des laufenden Subtests.

9.6.2.3. Subtest 1.2. Exception durch T-Bit-TRAP

Dieser Subtest testet den T-Bit-TRAP bei Vorgabe eines T-Bit im Prozessorstatuslangwort PSL <4> (Eintritt in den Kompatibilitätsmodus). Es werden zwei Tests durchgeführt (TST R2 und IOT); während TST R2 im Kompatibilitätsmodus unterbrechungsfrei abläuft, bewirkt der Befehl IOT seinerseits einen TRAP. Die Testschritte erfolgen analog dem Subtest 1.1.

9.6.2.4. Subtest 1.3. Exception durch die Befehle RTT und RTI

Dieser Subtest testet den TRAP durch die Befehle RTT und RTI bei Vorgabe eines T-Bit im Prozessorstatuswort PSW <4>, das im Kompatibilitätsmodus aus dem Stack (R6) geholt wird. Die Testschritte erfolgen analog dem Subtest 1.1.

9.6.2.5. Subtest 1.4. Exception durch ungerade Adresse

Dieser Subtest testet den Abbruch des Befehlsablaufes durch TRAP bei ungerader Adresse im Kompatibilitätsmodus.
Es werden drei Tests durchgeführt:

MOV mit ungerader Quelladresse
MOV mit ungerader Zieladresse
REI mit ungerader Rückkehradresse

Die Testschritte erfolgen analog dem Subtest 1.1.

9.6.2.6. Subtest 1.5. Exception bei fehlerhafter Befehlskodierung

Dieser Subtest testet den TRAP bei illegaler Befehlscodierung im Kompatibilitätsmodus. Es werden zwei Tests durchgeführt. JMP und JSR mit Registeradresse (dies ist verboten, da diese programmsteuernden Befehle eine Speicheradreßangabe benötigen). Die Testschritte erfolgen analog dem Subtest 1.1.

9.6.2.7. Subroutinen GEN/HNDLR

Diese Subroutinen sind zuständig dafür, Daten in den Stack abzulegen (GEN) und eine Exception im Kompatibilitätsmodus auszulösen (HNDLR). Die Generierung von GEN erfolgt in den fünf Subtests (9.6.2.2....9.6.2.6.), die Übernahme erfolgt bei Eintritt in die Routine. R8 enthält die Anfangsadresse der Tabelle GEN-BLK, die den folgenden Aufbau hat:

GEN-BLK: Ursprünglicher PC, der im Stack abgelegt ist (Adresse des SKR-Befehls, der ausgeführt wird).
GEN-BLK + 4: Ursprüngliches PSL, das im Stack abgelegt wird.
GEN-BLK + 8: Erwarteter PC nach Befehlsausführung.
GEN-BLK + 12: Erwartetes PSL nach Befehlsausführung.
GEN-BLK + 16: Codierung des unter Test stehenden TRAP-Befehls.

R9 enthält folgende Fehlerinformationen:

Bit0 = 1: fehlerhafte Codierung des TRAP-Befehls
Bit1 = 1: fehlerhafter PC
Bit2 = 1: fehlerhaftes PSL

9.6.2.8. Subroutine ERR RPT

Diese Routine protokolliert Fehlerausschriften mit Spaltenüberschriften. Durch einen CALL aufgerufen, werden jeweils vier Argumente (Fehlerausschriften) ausgegeben.

P1: Initialwerte der Quell-Daten
P2: Istwerte der Quell-Daten
P3: Initialwerte und Sollwerte der Ziel-Daten
P4: Istwerte der Ziel-Daten
Die Ausgabe erfolgt auf das Konsolterminal.

9.6.3. PVKAD2: SKR-Testroutinen für Test 1

Dieser Modul ist ausschließlich im SKR-Code programmiert und enthält die TRAP-Befehle und TRAP-auslösenden Befehlscodierungen wie sie, einschließlich ihrer Übergabe an das Testprogramm PVKAD1 durch die Routine GEN, in den Abschnitten 9.6.2.1...9.6.2.8. beschrieben sind.

9.6.4. PVKAD3: Teststeuerung Test 2, 3 und 4

Dieser Modul steuert die Tests 2, 3 und 4 des Kompatibilitätsmodus-Befehlstests, es erfolgt die in Abschnitt 9.5.1. beschriebene Testaufbereitung. Notwendige Voraussetzung ist das Vorhandensein des Diagnose-Supervisors.

9.6.4.1. Test 2 Steuerung des Tests der Einoperandenbefehle

1. Prüfen ob User-Modus, gegebenfalls Einstellung desselben.
2. Zugriffsflag auf "Schreiben" setzen.
3. Durchführung Test 2.
4. Zugriffsflag auf "Lesen" setzen.
5. 2.Durchführung Test 2.
6. Rückspeichern PSL -Testende-

9.6.4.2. Test 3-4 Steuerung des Tests der Zweioperandenbefehle bzw. Steuerbefehle

1. Prüfung ob User-Modus, gegebenfalls Einstellung desselben.
2. Durchführung Test 3 bzw. Test 4.
3. Rückspeichern PSL -Testende-

9.6.5. PVKAD4:Bedienungsroutinen für den Komatibilitätsmodus-Befehlstest

Dieser Modul ist eine allgemeine Bedienungsroutine für den Kompatibilitätsmodus-Befehlstest. Sie beinhaltet die Prüfung der Exception-Codes und den Aufruf der zugeordneten Bedienungsroutine, die Bereitstellung von Testdaten, als auch die Ausgabe von Fehlermitteilungen.

9.6.5.1. Allgemeine Bedienungsroutine für Kompatibilitätsmodus-Exceptions

Der Exception-Code wird geprüft und die zugeordnete Bedienungsroutine aufgerufen.
Die Exceptions werden wie folgt beantwortet:

RESET:	Aufsuchen und Speichern der ASCII-Codes des unter Test stehenden Kompatibilitätsmodus-Befehls für den Fehlerausdruck.
BPT:	Durchführung der entsprechenden Subtests.
IOT:	Beenden des Tests.
EMT:	Bereitstellen der initialisierten Daten und Bedingungs-Codes.
TRAP:	Prüfen der Daten und Ausgabe bei Fehler.

9.6.5.2. Bedienungsroutine "RESET"

1. Speichern ASCII-Code des unter Test stehenden Befehls.
2. Erhöhen PC.
3. RETURN zum Kompatibilitätsmodus.

9.6.5.3. Bedienungsroutine "BPT"

1. Erhöhen PC.
2. Aufruf der Subtest-Beendigungsroutine des Diagnose-Supervisors.
3. Erhöhen Subtestnummer.
4. Aufruf der Subtest-Eröffnugsroutine des Diagnose-Supervisors.
5. RETURN zum Kompatibilitätsmodus.

9.6.5.4. Bedienungsroutine "IOT"

1. Rückspeichern des PC.
2. Löschen des PSL.
3. Test fortführen und beenden.

9.6.5.5. Bedienungsroutine "EMT"

1. Bereitstellen aller Initial- und Sollwerte.
2. Setzen des Bedingungs-Code.
3. RETURN zum Kompatibilitätsmodus.

9.6.5.6. Bedienungsroutine "TRAP"

1. Prüfen aller Register, Daten und Bedingungs-Codes.
2. Im Fehlerfall Aufruf der Fehlerroutine des Diagnose-Supervisors.

9.6.5.7. TRAP bei fehlerhafter Adresse oder Codierung

1. fehlerhafte Adresse, Abbruch.
2. fehlerhafte Codes, Fehler.

9.6.5.8. Datenvorgabe für JMP und JSR

Diese Routine dient der Ermittlung und Bereitstellung der Daten für den Test der Befehle JMP und JSR. Sie bereitet einen Exception-Signalvektor auf und zeigt so auf den unter Test stehenden Befehl, bevor RETURN in den Kompatibilitätsmodus erfolgt.

Inhalt des Exception-Signalvektors:

SIG PC: EMT + Befehlstyp + Bedingungs-Code.
+2: PC-Sollwert nach Befehlsdurchführung.
+4: unter Test stehender Befehl.

9.6.5.9. Datenvorgabe für Einoperandenbefehle

Diese Routine dient der Ermittlung und Bereitstellung der Daten für den Test der Einoperandenbefehle. Sie bereitet einen Exception-Signalvektor auf und zeigt so auf den unter Test stehenden Befehl, bevor RETURN in den Kompatibilitätsmodus erfolgt.

Inhalt des Exception-Signalvektors:

SIG PC: EMT + Befehlstyp + Bedingungs-Code.
+2: Byteindikatoren bei Bytebefehlen + Tabellenlänge, wenn die Daten von der Tabelle geholt werden.
+4: Initial-Daten oder Initial-Daten-Tabellenadresse.
+6: Soll-Daten oder Soll-Daten-Tabellenadresse.
+8: unter Test stehender Befehl.

9.6.5.10. Datenvorgabe für Zweioperandenbefehle

Diese Routine dient der Ermittlung und Bereitstellung der Daten für den Test der Zweioperandenbefehle. Sie bereitet einen Exception-Signalvektor auf und zeigt so auf den unter Test stehenden Befehl, bevor RETURN in den Kompatibilitätsmodus erfolgt.

Inhalt des Exception-Signalvektors:

SIG PC: EMT + Befehlstyp + Bedingungs-Code.
+2: Byteindikatoren bei Bytebefehlen + Tabellenlänge, wenn die Daten von der Tabelle geholt werden.
+4: Initial-Daten für Zieladresse oder Initial-Daten-Tabellenadresse (Zieladresse).
+6: Soll-Daten für Zieladresse oder Soll-Daten-Tabellenadresse (Zieladresse).
+8: Initial-Daten für Quelladresse oder Initial-Daten-Tabellenadresse (Quelladresse).
+10: unter Test stehender Befehl.

9.6.5.11. Datenvorgabe Doppelwortbefehle

Diese Routine dient der Ermittlung und Bereitstellung der Daten für den Test von 32-Bit-Doppelwortbefehlen (SKR). Sie bereitet einen Exception-Signalvektor auf und zeigt so auf den unter Test stehenden Befehl, bevor RETURN in den Kompatibilitätsmodus erfolgt.

Inhalt des Exception-Signalvektors:

SIG PC: EMT + Befehlstyp + Bedingungs-Code.
+2: Byteindikatoren bei Bytebefehl + Tabellenlänge, wenn die Daten von der Tabelle geholt werden.
+4: Initial-Daten für NWT-Zieladresse oder Initial-Daten-Tabellenadresse (NWT-Zieladresse).
+6: Soll-Daten für NWT-Zieladresse oder Soll-Daten-Tabellenadresse (NWT-Zieladresse).
+8: Initialadresse für Quelladresse.
+10: Initial-Daten für HWT-Zieladresse oder Initial-Daten-Tabellenadresse (HWT-Zieladresse).
+12: Soll-Daten für HWT-Zieladresse oder Soll-Daten-Tabellenadresse (HWT-Zieladresse).
+14: unter Test stehender Befehl.

9.6.5.12. Datenvorgabe für STACK-MOVE

Diese Routine dient der Ermittlung und Bereitstellung der Daten für den Test der STACK-MOVE-Befehle. Sie bereitet einen Exception-Signalvektor auf und zeigt so auf den unter Test stehenden Befehl, bevor RETURN in den Kompatibilitätsmodus erfolgt.

Inhalt des Exception-Signalvektors:

SIG PC: EMT + Befehlstyp + Bedingungs-Code.
+2: Byteindikatoren bei Bytebefehl + Tabellenlänge, wenn die Daten von der Tabelle geholt werden.
+4: Initial-Daten oder Initial-Daten-Tabellenadresse.
+6: unter Test stehender Befehl.

9.6.5.13. Dateneintrag in Speicher und Register

Diese Routine stellt die Daten zusammen und speichert sie in spezifisch zugeordnete Speicherzellen und Register.
Dieses Übergabefeld gestaltet sich wie folgt:

SIG -Adresse des Exception-Signalvektors
MECH -Adresse des Exception-Funktionsvektors
REG_MODE -Registermodus des unter Test stehenden Befehls
REG_NUM -Registernummer des unter Test stehenden Befehls
IN_DATA_PNTR -Zeiger auf Initial-Daten im Speicher
EXP_DATA_PNTR -Zeiger auf Soll-Daten im Speicher
BYTE_INDCTR -Flag indizierender Bytebefehl
BUFF -Operandenadresse (Registermodus)
BUFPTR -Zeiger auf Operandenadresse (Registermodus)
INDX -Index (Registermodus)
RLTV -relative Operandenadresse (Registermodus)
TWNTSVN_PC -PC des Operanden (Immediatemodus)

Alle benötigten Daten werden bereitgestellt oder initialisiert, einschließlich der Soll-Resultate. Die Steuerung wird der rufenden Routine zurückgegeben und somit der Kompatibilitätsmodus-PC eingestellt. Abschließend erfolgt Rücksprung durch -RETURN-

9.6.5.14. Datenprüfung

Diese Routine gewährt, daß mittels der Initial-, Soll- und Ist-Daten, die in den zugeordneten Speicherplätzen gespeichert wurden (siehe auch 9.6.5.13.), alle Register, Daten und Bedingungs-Codes geprüft werden und im Fehlerfalle Fehlerermittlungen erfolgen.

9.6.5.15. Laden eines Registers

Diese Routine lädt und initialisiert ein angewähltes Register mit Initial-Daten und speichert die Soll-Daten für dieses Register.

9.6.5.16. Initialisieren aller Register

Diese Routine wird zu Testbeginn aufgerufen. Sie initialisiert alle Register und Bedingungs-Codes und speichert die Soll-Daten für diese Register.

9.6.5.17. Fehlerausgabe

Diese Routine ist eine allgemeine Fehlerausgaberoutine für alle Kompatibilitätsmodus-Befehle. Sie wird durch den Diagnose-Supervisor aufgerufen. Sie ermittelt im Fehlerfall den Befehlstyp und druckt die erforderlichen Kopfzeilen und Daten für diesen Befehl.

9.6.6. PVKAD5 SKR-Einoperandenbefehle

Dieser Modul enthält die Subroutinen für alle SKR-Einoperandenoperationen, die den wesentlichen Teil des SKR-Befehlssatzes darstellen.

9.6.6.1. Test 2 (Subtest 1...Subtest 360)

Diese Tests prüfen die Einoperandenbefehle entsprechend der zugeordneten Adressierung, wie sie in Abschnitt 9.5.13.1. als Test 2 aufgeführt sind. Im User-Modus werden alle Daten gesetzt einschließlich der Initialisierung aller Register und Bedingungsflags. Nach Ausführung werden alle erwarteten Daten, alle Register und alle Bedingungsflags geprüft. Soll-Ist-Abweichungen werden ausgegeben.

9.6.6.2. Darstellung des Testprinzips

Der Test eines Befehls "ABC" mit beliebiger Adressierung stellt sich in der Darstellung der Makro-Rufe wie folgt dar:

```
Zeilenzählwert_1   $ ERRMSG ABC
                     "
                     "

                 2   $ BGNSUB
                 3   SETUP WSO, INCC=..., INDST=..., EXDST=...,
-------------------------------
                 4   ABC...
                 5   CHECK...
```

Diese Makros sind Bestandteil des Diagnose-Supervisors. In aufgelöster Form, also in Darstellung der Makro-Körper, stellt sich der gleiche Test wie folgt dar:

```
Zeilenzählwert_1   0   000005   RESET
                   2   000003   .BYTE CNT
                   3   000101   .ASCII /ABC/
                   4   000102
                   5   000103
                2  6   000003   BPT
                3  10  104...   EMT <WSO *16 +INCC>
                   12  ......   .WORD ... ; INCC
                   14  ......   .WORD ... ; INDST
                   16  ......   .WORD ... ; EXDST
                4  20  ......   ABC ....
                5  22  104...   TRAP ...
```

zu_1: Aufbereitung der Fehlerausschrift durch Eintrag der Mnemonik für den unter Test stehenden Befehl. Diese Makroanweisung erfolgt jeweils nur beim erstmaligen Test eines jeden Befehls, das heißt nur bei sich ändernder Befehlsmnemonik (siehe auch 9.6.5.2.).

zu_2: Beenden des vorangegangenen und Eröffnen des folgenden Subtests (siehe auch 9.6.5.2.).

zu_3: Bereitstellung aller Initial- und Sollwerte, Setzen des Bedingungs-Code (siehe auch 9.6.5.5.).

zu_4: Durchführung des unter Test stehenden Befehls entsprechend der im Operandenfeld angegebenen Adressierungsart.

zu_5: Soll-Ist-Vergleich, Ausgabe von Fehlern (siehe auch 9.6.5.6.).

Der Test erfolgt zum Beispiel bei dem Befehl CLR durch einmalige Abarbeitung der unter Zeilenzählwert 1...5 bzw. 2...5 angeführten Schritte, bei der Mehrzahl der Befehle muß jedoch auch die Funktionsfähigkeit in Abhängigkeit vom Operanden geprüft werden. Bei diesen Tests werden die Schritte unter Zeilenzählwert 3...5 jeweils mit veränderten Initialwerten mehrfach wiederholt, bevor zur nächsten Adressierungsart oder zum nachfolgenden Befehl übergegangen wird.
Bis auf den unter Test stehenden Befehl handelt es sich hierbei ausschließlich um Exception-auslösende Anweisungen. Die zugeordneten Behandlungsprogramme sind in Abschnitt 9.6.5. beschrieben. Die Auswahl der für den unter Test stehenden Befehl erforderlichen Bedienungsroutine erfolgt durch nähere Interpretation des EMT.
Dies wird erreicht durch Angabe des Befehlstyps:

WSO = Wortbefehl-Einoperand
BSO = Bytebefehl-Einoperand
und einer Aussage über die Bedingungs-Codes.

Die aktuellen Werte für die Generierung der so ausgewählten Bedienungsroutine sind in dem jeweiligen Abschnitt des Prüfprogrammes angegeben.
Dabei bedeuten:

INCC = Initialwert Bedingungs-Code
INDST = Initialwert Zieladresse
EXDST = Sollwert Zieladresse

9.6.7. PVKAD6 SKR-Zweioperandenbefehle

Dieser Modul enthält die Subroutinen für alle SKR-Zweioperandenoperationen, die einen wesentlichen Teil des SKR-Befehlssatzes darstellen.

9.6.7.1. Test 3 (Subtest 1...Subtest 206)

Diese Tests prüfen die Zweioperandenbefehle entspechend der zugeordneten Adressierung, wie sie in Abschnitt 9.5.13.1. als Test 3 aufgeführt sind. Im User-Modus werden alle Daten gesetzt einschließlich der Initialisierung aller Register und Bedingungsflags. Nach Ausführung werden alle erwarteten Daten, alle Register und alle Bedingungsflags geprüft. Soll-Ist-Abweichungen werden ausgegeben.

9.6.7.2. Darstellung des Testprinzips

Das Testprinzip gleicht dem der Einoperandenbefehle. Der Test eines Befehls "ABC" mit beliebiger Adressierung stellt sich in der Darstellung der Makro-Rufe wie folgt dar:

```
Zeilenzählwert_1   $ ERRMSG ABC
                     "
                     "
                     "
                 2   $ BGNSUB
                 3   SETUP WDO, INCC=..., INSRC=..., EXDST=...
-------------------------------
                 4   ABC...
                 5   CHECK...
```

Aufgelöst in Makro-Körper ergibt dies folgende Darstellung.

```
Zeilenzählwert_1   0   000005   RESET
                   2   000003   .BYTE CNT
                   3   000101   .ASCII /ABC/
                   4   000102
                   5   000103
                 2 6   000003   BPT
                 3 10  104...   EMT <WDO *16 +INCC>
                   12  ......   .WORD ... ; INCC
                   14  ......   .WORD ... ; INDST
                   16  ......   .WORD ... ; EXDST
                   20  ......   .WORD ... ; INSRC
                 4 22  ......   ABC ....
                 5 24  104...   TRAP ...
```

Die Makros sind Bestandteil des Diagnose-Supervisors. Die zugeordneten Exception-Behandlungsprogamme sind in Abschnitt 9.6.5. beschrieben. Die Auswahl der für den unter Test stehenden Befehl erforderlichen Bedienungsroutine erfolgt durch nähere Interpretation des EMT.
Dies wird erreicht durch Angabe des Befehlstyps:

WDO = Wortbefehl-Doppeloperand
BDO = Bytebefehl-Doppeloperand
und einer Aussage über die Bedingungs-Codes.

Die aktuellen Werte für die Generierung der so ausgewählten Bedienungsroutine sind in dem jeweiligen Abschnitt des Prüfprogrammes angegeben (siehe Programmassemblerausdruck).
Dabei bedeuten:

INCC = Initialwert Bedingungs-Code
INDST = Initialwert Zieladresse
EXDST = Sollwert Zieladresse
INSRC = Initialwert Quelladresse

9.6.8. PVKAD7 Steuerbefehle

Dieser Modul enthält die Subroutinen für alle SKR-Steuerbefehle.

9.6.8.1. Test 4 (Subtest 1...Subtest 46)

Diese Tests prüfen die Steuerbefehle. Im User-Modus werden alle Daten gesetzt einschließlich der Initialisierung aller Register und Bedingungsflags. Nach Ausführung werden alle erwarteten Daten, alle Register und Bedingungsflags geprüft. Soll-Ist-Abweichungen werden ausgegeben.

9.6.8.2. Subtest 1 Darstellung des Testprinzips für den BRANCH-Befehl

```
          $ ERRMSG BR
          $ BGNSUB
-----------------
          SETUP BRANCH, INCC=..., INDST=DST2
          BR DST2           ; BR VORWÄRTS
          CHECK             ; FEHLER
----------------
  DST1 : BR 1$
----------------
  DST2 : SETUP BRANCH, INCC=..., INDST=DST1
          BR DST1           ; BR RÜCKWÄRTS
          CHECK             ; FEHLER
----------------
    1$ : BR     OFF100
OFF277 : BR     OFF40
OFF337 : BR     OFF20
OFF357 : BR     OFF10
OFF367 : BR     OFF4
OFF373 : BR     OFF2
OFF375 : BR     OFF0
----------------
OFF  0 : BR     DONEBR
----------------
OFF  2 : BR     OFF375    ; OFFSET + 2
         CHECK            ; FEHLER
OFF  4 : BR     OFF373    ; OFFSET + 4
         CHECK            ; FEHLER
OFF 10 : BR     OFF367    ; OFFSET +10
         CHECK            ; FEHLER
OFF 20 : BR     OFF357    ; OFFSET +20
```

```
          CHECK              ; FEHLER
OFF 40 : BR     OFF337       ; OFFSET +40
          CHECK              ; FEHLER
OFF100 : BR     OFF277
          CHECK              ; FEHLER
DONEBR : ...
```

Zunächst wird also der BRANCH vorwärts und rückwärts prinzipiell getestet, und anschließend erfolgt die Modifikation des Offset in den Schritten 100, 40, 20, 10, 4, 2, 0 wiederum vorwärts und rückwärts.
Die Makros sind Bestandteil des Diagnose-Supervisors. Die Exception-Behandlungsprogramme sind in Abschnitt 9.6.5. beschrieben. Die Auswahl der erforderlichen Bedienungsroutinen erfolgt durch nähere Interpretation des EMT im Makro SETUP (siehe auch Abschnitt 9.6.6.2.).
Dabei bedeuten:

```
BRANCH = BRANCH-Befehl
INCC   = Zustand der Bedingungsflags
INDST  = Zieladresse=Sprungadresse
```

9.6.8.3. Subtest 2 Darstellung des Testprinzips am Beispiel BMI, BPL

```
          $ BGNSUB
          $ ERRMSG BMI
          SETUP BRANCH, INCC=..., INDST=DST1
-----------------
          BMI 1$             ; BEDINGUNG NICHT ERFÜLLT
  DST1 : BR  2$
-----------------
     1$ : CHECK              ; FEHLER
-----------------
     2$ : $ ERRMSG BPL
          SETUP BRANCH, INCC=..., INDST=DST2
-----------------
          BPL DST2           ; BEDINGUNG ERFÜLLT
          CHECK              ; FEHLER
-----------------
  DST2 : ...
```

Dieser Test weist die prinzipielle Abhängigkeit der bedingten BRANCH-Befehle von den Bedingungsflags nach.
Mit nicht erfüllter Bedingung sind in den Test einbezogen die Befehle BMI, BEQ, BVS, BCS, und mit erfüllter Bedingung die Befehle BPL, BNE, BVC und BCC.
Die Makros sind Bestandteil des Diagnose-Supervisors. Die zugeordneten Exception-Behandlungsprogramme sind in Abschnitt 9.6.5. beschrieben. Die Auswahl der erforderlichen Bedienungsroutinen erfolgt durch nähere Interpretation des EMT im Makro SETUP (siehe auch Abschnitt 9.6.6.2.).
Dabei bedeuten:

```
BRANCH = BRANCH-Befehl
INCC   = Zustand der Bedingungsflags
INDST  = Zieladresse (bei erfüllter Bedingungs-Sprungadresse)
```

9.6.8.4. Subtest 3...Subtest 20 Darstellung des Testprinzips am Beispiel BNE

```
            $ BGNSUB
            $ ERRMSG BNE
            SETUP BRANCH, INCC=..., INDST=DST1
----------------
            BNE 1$            ; BEDINGUNG NICHT ERFÜLLT
----------------
      DST1 : SETUP BRANCH, INCC=..., INDST=DST2
----------------
            BNE DST2          ; BEDINGUNG ERFÜLLT
----------------
         1$ : CHECK           ; FEHLER
----------------
      DST2 : ...
```

Es werden alle bedingten BRANCH-Befehle bei erfüllter und nicht erfüllter Sprungbedingung geprüft.
Die Makros sind Bestandteil des Diagnose-Supervisors. Die zugeordneten Behandlungsprogramme sind in Abschnitt 9.6.5. beschrieben. Die Auswahl der erforderlichen Bedienungsroutinen erfolgt durch nähere Interpretation des EMT im Makro SETUP (siehe auch Abschnitt 9.6.6.2.).
Dabei bedeuten:

```
BRANCH = BRANCH-Befehl
INCC   = Zustand der Bedingungsflags
INDST  = Zieladresse (bei erfüllter Bedingungs-Sprungadresse)
```

9.6.8.5. Subtest 21...Subtest 43 Darstellung des Testprinzips am Beispiel JMP

```
            $ ERRMSG JMP
            $ BGNSUB
            SETUP IJMP, INCC=..., INDST=DST1
----------------
            JMP...
----------------
            CHECK             ; FEHLER
----------------
      DST1 : ...
```

Die Befehle JMP, JSR und RTS sind in ihrer Funktion weder abhängig von den Bedingungsflags noch setzen sie sie. Um für alle Adressierungsarten diese Unabhängigkeit von den Bedingungsflags nachzuweisen, erfolgt der Test zweimal, mit gesetzten und gelöschten Bedingungsflags.
Für den Befehl JSR wird darüberhinaus in der Adressierungsart Modus 3, Register 7 der Test für alle Register R0 bis R5 durchgeführt. Ebenfalls erfolgt für alle Register der Test des RTS.
Die Makros sind Bestandteil des Diagnose-Supervisors. Die zugeordneten Behandlungsprogramme sind im Abschnitt 9.6.5. beschrieben. Die Auswahl der erforderlichen Bedienungsroutinen erfolgt durch nähere Interpretation des EMT im Makro SETUP (siehe auch Abschnitt 9.6.6.2.).

Dabei bedeuten:

IJMP = JMP-Befehl (I=Instruktion)
INCC = Zustand der Bedingungsflags
INDST = Zieladresse=Sprungadresse

9.6.8.6. Subtest 44 Darstellung des Testprinzips für den Befehl SOB

```
        $ ERRMSG SOB
        $ BGNSUB
        SETUP ISOB, INCC=..., INDST
----------------
        CLR R2
    1$ : INC R2
        CMP #, R2
        CCC
        SOB R1,1$
    2$ : CHECK
```

Der Befehl SOB wird mit allen Registern getestet. Der Test setzt die Funktionstüchtigkeit der Befehle CLR, INC, CMP, BMI und CCC voraus.
Die Makros sind Bestandteil des Diagnose-Supervisors. Die zugeordneten Behandlungsprogramme sind in Abschnitt 9.6.5. beschrieben. Die Auswahl der erforderlichen Bedienungsroutinen erfolgt durch nähere Interpretation des EMT im Makro SETUP (siehe auch Abschnitt 9.6.6.2.).
Dabei bedeuten:

ISOB = Befehl SOB
INCC = Zustand der Bedingungsflags
INDST = Zieladresse=Sprungadresse

9.6.8.7. Subtest 45 Darstellung des Testprinzips für die Bedingungsflag-Behandlungs-Befehle

```
        $ BGNSUB
        $ ERRMSG ABC
        SETUP SETCC, INCC=...,
----------------
        ABC
        CHECK
```

Der Test der Bedingungsflag-Behandlungs-Befehle "ABC" läuft nach dem angeführten Testprinzip nacheinander für die Befehle CCC, SCC, CLC, CLV, CLZ, CLN, SEC, SEV, SEZ und SEN ab.
Die Makros sind Bestandteil des Diagnose-Supervisors. Die zugeordneten Behandlungsprogramme sind in Abschnitt 9.6.5. beschrieben. Die Auswahl der erforderlichen Bedienungsroutinen erfolgt durch nähere Interpretation des EMT im Makro SETUP (siehe auch Abschnitt 9.6.6.2.).
Dabei bedeuten:

SETCC = Bedingungsflag-Behandlungs-Befehl
INCC = Zustand der Bedingungsflags

9.6.8.8. Subtest 46 Darstellung des Testprinzips für den Befehl NOP

```
            $ BGNSUB
            $ ERRMSG NOP
            SETUP SETCC, INCC=...,
      ----------------
            NOP
            CHECK
```

Dieser Test erfolgt zweimal, mit gelöschten und mit gesetzten Bedingungsflags. Die Makros sind Bestandteil des Diagnose-Supervisors. Die zugeordneten Behandlungsprogramme sind in Abschnitt 9.6.5. beschrieben. Die Auswahl der erforderlichen Bedienungsroutinen erfolgt durch nähere Interpretation des EMT im Makro SETUP (siehe auch Abschnitt 9.6.6.2.).
Dabei bedeuten:

SETCC = NOP-Befehl
INCC = Zustand der Bedingungsflags

9.7. Abkürzungsverzeichnis

Alle Abkürzungen für Befehle, Pseudo-Befehle, spezielle Register, Adreßbezeichnungen u.a. sind der Systemdokumentation zu entnehmen, bzw. erklären sich an der Befehlsbeschreibung in Abschnitt 9.5.
Für Abkürzungen, die als Makro-Namen auftreten, wird auf die Beschreibung des Diagnose-Supervisors verwiesen, bzw. sie erklären sich an der Funktionsbeschreibung der Makros in Abschnitt 9.6.6.2. Alle weiteren Abkürzungen innerhalb dieser Programmbeschreibung sind in den Abschnitten beschrieben, in denen sie auftreten.

9.8. Auflistung der Tests

Das Programm ist in vier Testkomplexe gegliedert.

Test 1 : Test der Exceptions
Test 2 : Test der Einoperandenbefehle
Test 3 : Test der Zweioperandenbefehle
Test 4 : Test der Steuerbefehle

10. PVKAE Test der privilegierten Architekturbefehle

10.1. Zusammenfassung

Das Diagnoseprogramm PVKAE, Dok.-Nr. 1.57.700006.7, ist ein Programm der Ebene 3, d.h. es läuft unter Steuerung des Diagnose-Supervisors, PSSAA, im stand-alone-Modus.
Das Programm meldet sich auf dem Terminal mit " RVS Privileged Architecture Exerciser ".
Es testet die wesentlichsten Teile der K1840-Architektur, die Privilegien erfordern, die über die von Nutzerprogrammen im User-Modus unter SVP hinausgehen. Dazu gehören die Überprüfung der richtigen Ausführung von Traps, Interrupts und Exeptions einschließlich der dazu gehörenden Return-Befehle, das Testen der Privilegierten Befehle, wie Change-Mode- ,Prozessorregister-, PROBE- und Prozesscontext-Befehle sowie die Funktionsprüfung des Intervallzeitgebers, der Speicherverwaltung und der Stackoperationen.
Dieses Testprogramm läuft unter Steuerung des Diagnose-Supervisors (PSSAA) im Standalone-Modus (d.h. dieser muß direkt von der FSE oder vom FPS gebootet und darf nicht vom SVP aus aufgerufen werden , Level 3). Soweit als möglich werden die Eigenschaften des Diagnose-Supervisors (DS), d.h. z.B. die Teststeuerkommandos, Organisation der Fehlerausschriften usw. genutzt.
Das Programm dient mit zur Überprüfung der Gesamtfunktion der CPU einschließlich Hauptspeicher.

10.2. Testbedingungen

10.2.1. Hardware

Beliebige K1800-Rechnerkonfiguration (CPU K2810 einschließlich Hauptspeicher K3581)

10.2.2. Software

Diagnose-Supervisor PSSAA im Standalone-Modus

10.3. Voraussetzungen

- Fehlerfreie Abarbeitung der CPU-Mikrodiagnose, des Grundspeichertestes PVKAS und des Grundbefehltestes PVKAA
- Diagnose-Supervisor im Standalone-Modus

10.4. Bedienungshinweise

Um den DS zu laden und zu starten, sowie diesen Test vom DS aus zu laden und unter Steuerung des DS abzuarbeiten, ist von der Bedienungsanleitung für den DS auszugehen.

10.4.1. Laden und Starten

Vor dem Start des Programmes ist der in dem Rechnersystem vorhandene CPU-Typ einzubinden (attach) und für den Test logisch auszuwählen (select).

```
>>>B SU
```

Es meldet sich der Diagnosesupervisor.

```
DS> ATTACH KAA40 HUB KA0 Y Y 17FF 1
DS> SEL KA0
DS> R PVKAC
```

10.5. Funktionsbeschreibung

10.5.1. Übersicht

Siehe Pkt. 10.1

10.5.2. Programmgröße

Das Programm erfordert einen Speicherplatz von etwa 105 Speicherseiten, das entspricht etwa 54 kByte.

10.5.3. Programmlaufzeit

Ein fehlerfreier Programmdurchlauf erfolgt in etwa 40 Sekunden.

10.5.4. Abarbeitungsbedingungen

Entfällt

10.5.5. Flag-Verwendung

Entfällt

10.5.6. Allgemeine Programmanforderungen

Entfällt

10.5.7. Fehlerermittlung

Der DS erzeugt bei erkanntem Fehler eine Standard-Fehlerkopf-Ausschrift, die die Programmidentifikation, die Test-, Subtest- sowie Fehler-Nr., Fehlerklasse, das getestete Gerät (KA0) und eine dem Test zugeordnete Ausschrift über die ausgeführte Funktion enthält.

Beispiel:

```
************ RVS Privileged Architecture Exerciser PVKAE - 1.0 ************
PVKAE V 1.0
PASS 1    TEST 2    SUBTEST 3      ERROR 1     29-SEP-1986      09:03:13,42
HARD ERROR WHILE TESTING KA0 : (verbale Beschreibung des Fehlers)
(Zusatzinformationen, wie z.B. :
- aktueller Registerinhalt (z.B. PSL, ICR, usw.)
- erwarteter Registerinhalt )

******************** End of Hard Error number 2  ************************
```

Ausgehend von der ausgeschriebenen Test-, Subtest- und Fehler- Nummer sowie der verbalen Beschreibung des Fehlers ist unter Zuhilfenahme der Programmassemblerlisten der Fehler zu analysieren und unter Verwendung der Stromlaufpläne der in Frage kommenden Funktionsgruppen durch entsprechende Messungen auf der betreffenden Steckeinheit der Fehler zu lokalisieren und zu beseitigen.

10.5.8. Programmitteilungen

Entfällt

10.5.9. Leistungsparameter des Testobjektes

Entfällt

10.5.10. Aktionen bei unerwarteten Fehlern

Entfällt

10.5.11. Testfolge bei mehreren Einheiten

Entfällt

10.5.12. Einstellung für Ablauf unter AFT

Entfällt.

10.5.13. Testsektionen

10.5.13.1. Tests und Subtests

Das Gesamttestprogramm wird in folgende Einzeltests unterteilt:

Test Nr.	1:	Interrupt-Test
	2:	Exception-Test
	3:	REI-Befehlstest
	4:	Test der Change-Mode-Befehle
	5:	Test der Prozessorregister-Befehle
	6:	Test des Intervallzeitgebers
	7:	Test der Speicherverwaltung (MM)
	8:	Test "Systemstack nicht gültig"
	9:	CHMx-Befehle zu schreibgeschützten Stacks
	10:	Test der PROBE-Befehle
	11:	Test der Prozeßcontext-Befehle

Durch entsprechende Supervisor-Kommandos beim Starten des Programms können diese Tests einzeln oder in Gruppen, einmalig oder als Dauertest beliebig lange oder mit einer vorgegebenen Anzahl von Durchläufen abgearbeitet werden (s.Kapitel 1, Beschreibung DS).
Diese Tests werden weiter untergliedert in Subtests. Inhalt und prizipielle Ausführung der Tests und Subtests sind unter Punkt 10.6. beschrieben.

10.5.13.2. Testsektionen

Bestimmte Tests sind unter inhaltlichen Gesichtspunkten zu Sektionen jeweils zusammengefaßt und können unter den nachfolgenden Namen aufgerufen und abgearbeitet werden:

DEFAULT	Durchlauf aller Tests
MEM MGT	Test 7,8,9 und 10
EXCEPTIONS	Test 1,2,3,8 und 9
PROCESS	z.Zt. keine Tests
REGISTERS	Test 1,5 und 6
CHANGE MODE	Test 4 und 9
TIMER	Test 6
CONTEXT	Test 11

10.6. Testbeschreibungen

10.6.1. Test 1 , Softwareinterrupts

Dieser Test prüft die K1840-Softwareinterruptlogik ab. Die Hardwareinterruptlogik wird hier nicht gesondert geprüft, jedoch in Verbindung mit der Hardware, die die Interrupts erzeugt, indirekt mit abgetestet.

10.6.1.1. Subtest 1.1 Software-Interrupt-Summary-Register (SISR)

Dieser Subtest prüft, ob durch Setzen der entsprechenden Bits im SISR (ID:0E, PR:15 auf ICL, BLP 8915) Softwareinterrupts ausgelöst werden und ob anstehende Softwareinterrupts im SISR erkennbar sind und durch Löschen der entsprechenden Bits im SISR zurückgesetzt werden können.

Prüfschritte:

1. Übergang zum Kernel-Modus ; IPL=0
2. Lesen des SISR
3. Prüfen, ob Bit 1 ...15 gelöscht sind
4. Setzen SISR
5. Prüfen der richtigen Interruptausführung
6. Prüfen, ob SISR=0
7. Prüfen IPL
8. Setzen SISR ; kein Interrupt
9. Prüfen SISR = 1
10. Löschen SISR
11. Prüfen SISR = 0
12. REI
13. Setzen SISR [15:00] ; Interrupt auf Niveau 15 ?
14. Prüfen auf richtige Interruptausführung
15. Prüfen SISR
16. Prüfen IPL
17. REI
18. Abarbeiten der Pkte. 14 bis 17 jeweils für das nächst niedrige Niveau bis SISR = 0
19. Prüfen SISR = 0

Mögliche Fehler: - unerwartete Exceptions oder Interrupts - Softwareinterrupts nicht richtig ausgeführt

10.6.1.2. Subtest 1.2 Software-Interrupt-Request-Register (SIRR)

Dieser Subtest prüft die Erzeugung von Softwareinterrupts durch Eintragen des Niveaus (level) in das SIRR (PR:04, auf ICL, BLP 8915).

Prüfschritte:

1. Übergang zum Kernel-Modus ; IPL = 0. Die Punkte 2 bis 8 werden dann jeweils für I = 1 bis 15 abgearbeitet:
2. Laden SIRR mit I ; Interrupt ?
3. Prüfen der richtigen Interruptausführung
4. Prüfen SISR = 0 ?
5. Prüfen IPL
6. LADEN SIRR mit I ; kein Interrupt !
7. Prüfen SISR = 1 ?
8. REI
9. SIRR laden mit FFFFFFF0 hex., d.h. SIRR [03:00]=0; kein Interr.
10. Prüfen SISR = 0 ?

Mögliche Fehler wie bei Subtest 1.1

10.6.2. Test 2 - Exceptions

Dieser Test prüft teilweise die K1840-Exceptionlogik. Die meisten Teile dieser Logik werden nicht direkt, sondern in Verbindung mit der Logik geprüft, die die Exceptions erzeugt.

10.6.2.1. Subtest 2.1 Exceptions bei reservierten Befehlen

Dieser Subtest prüft vollständig in allen Modi die bei Verwendung reservierter Befehle zu erwartenden Exceptions. Dabei wird der für alle Zeiten reservierte Oprationskode FFFF hex. verwendet.

Prüfschritte:

1. Operationskode FFFF im Kernelmodus
2. Operationskode FFFF im Executivemodus
3. Operationskode FFFF im Supervisormodus
4. Operationskode FFFF im Usermodus

Es wird jeweils die richtige Ausführung einer Exception sowie der Prozess-Status abgeprüft. Es erfolgt eine Modifizierung des Rückkehr-PC und Ausführung eines REI. Dieser Test prüft vollständig die richtige Ausführung des REI-Befehls, um zu sichern, daß eine exakte Fehlerbehandlung bei allen reservierten Operanden sowie bei Änderung von Modus und Stack erfolgt.

10.6.2.2. Subtest 2.2 Exceptions bei privilegierten Befehlen

Dieser Subtest prüft, ob alle privilegierten Befehle (HALT, LDPCTX, SVPCTX, MTPR und MFPR) im User-, Supervisor- und Executive-Modus zu Exceptions führen, wobei der Prozeßstatus abgeprüft wird. Es wird dann jeweils der Rückkehr-PC modifiziert mit nachfolgendem REI. Bei fehlender oder fehlerhafter Exception erfolgt Fehlerausschrift und Beendigung des Testes.

10.6.2.3. Subtest 2.3 Exceptions bei reservierten Adressierungsmodi

Dieser Subtest prüft ab, ob reservierte Adressierungsmodi fehlerfrei zu den entsprechenden Exceptions führen, ansonsten erfolgt Fehlerausschrift und Testbeendigung. Im einzelnen werden folgende Fälle abgeprüft:

- short literal — Schreiben, Modifizieren, Adressieren, auch innerhalb eines Index
- register: — Adresse, auch innerhalb eines Index
- index mode: — PC als Index oder innerhalb eines Index

- CLRB mit
  - short literal , write
  - short literal , modify
  - short literal , address
  - short literal , field
  - indexmode und indexmode
  - PC als Indexregister

- Register als address-source
- Register innerhalb eines Indexmodus

Es wird jeweils die Exceptioninformation geprüft, der Rückkehr-PC modifiziert und ein REI ausgeführt.

10.6.3. Test 3 - REI-Befehlstest (return from exception or interrupt)

10.6.3.1. Subtest 3.1 Keine Exceptions oder Interrupts

Es wird der REI-Befehl im fehlerfreien Modus abgeprüft, um zu kontrollieren, ob Änderungen von Modus und Stack richtig ausgeführt werden.
Eine allgemeine Routine greift auf eine Tabelle mit den Testparametern, Anfangs-PSL und Test-PSL für alle Testfälle zurück. Eine BPT-Exception initialisiert den IPL auf 31 im Interruptstack. Ein REI setzt dann das Anfangs-PSL. Ein zweiter REI holt letztlich das Test-PSL. Nach jedem Schritt erfolgen entsprechende Prüfungen. Dieser Ablauf erfolgt fuer 31 verschiedene Testfälle (s. Listings), wobei jeweils folgende Prüfschritte abgearbeitet werden:

1. Setzen eines Handlers für unerwartete Exceptions und Interrupts
2. Setzen IS = 1 und IPL = 31
3. Holen des Vektors für reservierte Operationskodes
4. Holen des Vektors für AST, Niveau 2
5. Holen des Vektors für reservierte Operanden
6. Holen des Vektors für Kompatibilitätsmodus

10.6.3.2. Subtest 3.2 Exceptions bei reservierten Operanden

Dieser Subtest prüft die richtige Ausführung von Exceptions bei Verwendung reservierter Operanden, wobei der prinzipielle Testablauf für 43 verschiedene Testfälle der gleiche ist, wie bei Subtest 3.1

10.6.3.3. Subtest 3.3 Trace-Traps

Der allgemeine Testablauf für 7 verschiedene Testfälle ist wiederum der gleiche wie bei Subtest 3.1 , jedoch mit folgenden Einzeltestschritten:

1. Initialisierung durch Setzen von IS=1 und IPL=31
2. Teilschritte für jeden Testfall:
   2.1 PUSH Test-PSL
   2.2 PUSH PC=REI-PC-2
   2.3 Setzen Trace-Trap-Vektor
   2.4 Im PSW Setzen der Bits "initial trace pending" und "trace trap"
   2.5 REI

10.6.3.4. Subtest 3.4 Asynchrone Systemtraps (AST)

Dieser Subtest prüft unter Verwendung des REI asynchrone Systemtraps im fehlerfreien Modus im Niveau 0 bis 3. Dabei wird abgetestet, ob Modus und Stack richtig geändert werden und nur Interrupts bei IPL Niveau 2 auftreten. Der sonstige

prinzipielle Testablauf erfolgt wie bei Subtest 3.1 dargestellt.

10.6.4. Test 4 - Test der Change-Mode-Befehle CHMx

Dieser Test prüft die richtige Änderung von Modus und Stack durch die Befehle CHMK, CHME, CHMS und CHMU. Er ist nicht in Subtests untergliedert.

Die prinzipiellen Testschritte sind:

1. Initialisierung
2. Setzen der Handler für Exceptions u. Interrupts
3. Setzen IS=1 und IPL=31
4. Holen der CHMx-Vektoren
5. Ablauf für die einzelnen Befehle jeweils:

   5.1 REI-PC-1 ---> stacked PC
   5.2 initial PSL ---> stacked PSL
   5.3 REI

10.6.5. Test 5 - Prozessorregisterbefehle

Dieser Test prüft die richtige Ausführung der Befehle MTPR und MFPR. Dabei wird von der fehlerfreien Abarbeitung der Hardcore-Tests in der Mikrodiagnose ausgegangen.

10.6.5.1. Subtest 5.1 Test des PME-Registers

Das Einbitregister PME (performance monitor enable), im CES (ID:0C, PR:13, auf ICL, BLP 8915), wird durch einen MTPR zum SCB beschrieben, dann durch einen MFPR gelesen mit anschließendem Soll-/Istwert-Vergleich. Vor diesem Subtest wird PME gerettet und anschließend rückgespeichert.

10.6.5.2. Subtest 5.2 Test des SID-Registers (system identific.)

Dieser Subtest prüft ab, ob das SID-Register (ID:03, PR:3E, auf CIB, BLP 8929) gelesen werden kann und ob beim Versuch, es zu beschreiben ein "reserved operand fault" ausgelöst wird.

Prüfschritte:

1. MFPR vom SID
2. Soll-/Istwert-Vergleich
3. Holen des Vektors "reserved operand"
4. MTPR von -1 zum SID
5. Prüfen, ob "reserved operand exception"
6. Return

10.6.5.3. Subtest 5.3
Fault "reservierter Operand"

Es wird geprüft,ob beim Schreiben und Lesen (MTPR und MFPR) bzgl. eines nichtexistierenden Registers (Adr. 07FFFFFF hex) ein "reserved operand fault" ausgelöst wird.

Prüfschritte:

1. Holen des Vektors "reserved operand"
2. Befehl MTPR #0,#X7FFFFFF
3. Prüfung der "reserved operand data"
4. Modifizierung der Rückkehradresse , REI
5. entsprechender Ablauf für MFPR

10.6.6. Test 6 - Test des Intervalltimers

Dieser Test prüft alle Funktionen und Register des Intervallzeitgebers und, soweit in der Anlage vorhanden, die Echtzeituhr.

10.6.6.1. Subtest 6.1
Interrupt-enable-Bit

Prüfung des Setzens und Löschens des Interrupt-enable- Bits (Bit 06) im Kommando/Status-Register des Intervallzeitgebers (interval clock control/status) ICCS (ID:0A ,PR:18, auf ICL, BLP 8915).

Prüfschritte:

1. Setzen Interruptenable-Bit , Lesen des ICCS
2. Löschen ICCS , Lesen ICCS

10.6.6.2. Subtest 6.2
Test des XFR- (XEGR-) Bits

Es wird überprüft, ob durch Setzen des XFR-Bits (Bit 04 im ICCS) ein Transport des Inhaltes des Registers "nächstes Intervall" (next interval count register) NICR (ID:09, PR:19, auf CEH, BLP 8914 und ICL, BLP 8915) in das aktuelle Intervallregister (interval count register) ICR (ID:0B, PR:1A , auf CEH und ICL) aktiviert wird und ob das w/o-XFR-Bit als "0" gelesen wird.

Prüfschritte:

1. Laden des NICR mit "alles 1"
2. Setzen des XFR-Bits im ICCS
3. Prüfen, ob XFR-Bit als "0" gelesen wird
4. Prüfen, ob durch Transport NICR ---> ICR im ICR "alles 1"

10.6.6.3. Subtest 6.3
Einzeltakt für ICR

Dieser Subtest überprüft, ob durch Setzen des w/o-Einzeltaktbits (SGLCLK, single clock, Bit 05) im ICCS eine Inkrementierung des Inhaltes von ICR erfolgt und ob dieses Bit als "0" gelesen wird.

Prüfschritte:

1. Laden des NICR mit einem Wert, der 2 Mikrosekunden entspricht.
2. Transfer NICR nach ICR durch Setzen des XFR-Bits.
3. Setzen SGLCLK-Bit
4. Prüfen, ob SGLCLK-Bit als "0" gelesen wird
5. Prüfen, ob Inhalt von ICR "alles 1"

10.6.6.4. Subtest 6.4 ICR-Test

Das ICR wird mit 9 verschiedenen Datenmustern getestet, indem diese Muster zunächst in einem allgemeinen Register (R3) erzeugt , dann in das NICR geladen und mittels XFR in das r/o-ICR transferiert und dessen Inhalt dann mit dem Ausgangsmuster jeweils verglichen werden.

Datenmuster:
- AAAAAAAA
- 55555555
- 33333333
- 0F0F0F0F
- 00FF00FF
- 0000FFFF
- FFFF0000
- FF00FF00
- F0F0F0F0

10.6.6.5. Subtest 6.5 Intervalltaktfolge

Von diesem Subtest wird die richtige Zählfolge des ICR durch Einzeltakte, d.h. Setzen des SGLCLK-Bits im ICCS, überprüft.
Prüfschritte:

1. Laden der allgemeinen Register zum Erzwingen von C- (carry) Bits
1.1 R6 = erwartete C-Bitposition
1.2 R2 = C-Bitzähler
1.3 R3 = Wert zum Erzwingen des C-Bits
2. R3 ---> NICR
3. NICR ---> ICR
4. Setzen SGLCLK in ICCS, um das erwartete C-Bit zu erzwingen und zu kontrollieren
5. Inkrementieren von R2 bis 31, d.h. 32 Testdurchläufe

10.6.6.6. Subtest 6.6 Intervall-Clock-Überlauf

Es wird abgetestet, ob das Fehlerbit (Bit 31) im ICCS gesetzt wird, wenn ein Überlauf des ICR auftritt bevor ein vorheriger Interrupt bedient wurde.

Prüfschritte:

1. Laden NICR mit "alles 1"
2. NICR ---> ICR mittels XFR

3. Einzeltakt durch SGLCLK, damit Überlauf provoziert
4. Wiederholung 2. und 3.
5. Prüfen, ob sowohl Fehler- als auch Interruptanforderungsbit im ICCS gesetzt sind

10.6.6.7. Subtest 6.7 Intervall-Clock ohne Interrupt

Es wird die Funktion des RUN-Bits (Bit 00) im ICCS geprüft, ohne daß das Interrupt-enable-Bit (Bit 06) im ICCS gesetzt ist, also kein Interrupt zugelassen ist.

Prüfschritte:

1. Laden NICR mit einer Größe, die etwa 1 ms entspricht
2. NICR ---> ICR mittels XFR
3. Laden NICR mit "alles 0"
4. Setzen RUN-Bit im ICCS
5. Warten auf INT-Bit (Interruptanforderung, Bit 07)
6. Rücksetzen RUN-Bit
7. Rücksetzen Interruptanforderung durch Setzen Bit 07 im ICCS
8. Prüfen, ob Clock nach Interruptanforderung weiterzählt
9. Fehlerbit (Bit 31) gesetzt ?
10. Läßt sich Fehlerbit löschen ?

10.6.6.8. Subtest 6.8 Intervall-Clock mit Interrupt

Es wird die Funktion des RUN-Bits im ICCS bei gesetztem IE-Bit abgetestet.

Prüfschritte:

1. Laden NICR mit einem Wert, der etwa 40 Mikrosek. entspricht.
2. NICR ---> ICR mittels XFR
3. Laden NICR mit "alles 0"
4. Laden der Adresse des Intervalltimer-Handlers in Vektor C0 hex.
5. Herabsetzen der CPU-Priorität
6. Setzen IE-Bit und RUN-Bit im ICCS
7. Stop der CPU für 20 ms. Während dieser Zeit muß ein Timerinterrupt erfolgen, ansonsten Fehlerausschrift
8. Rücksetzen INT und RUN
9. Prüfen, ob Interrupt auf IPL = 24 ?
10. Fehlerbit im ICCS prüfen

10.6.6.9. Subtest 6.9 Weiterzählen des TODR (time of day register)

Es wird die Funktion des time of day registers TODR (ID:01, PR:1B, auf IRC, BLP 8908) bezüglich des Weiterzählens von einem bekanntem vorgegebenem Wert aus abgetestet.

Prüfschritte:

1. Setzen eines Verzögerungszählers
2. Laden TODR mit FFFF1944 hex.
3. Lesen TODR
4. Wenn TODR weitergezählt hat, Übergang zum nächsten Subtest, ansonsten timeout abwarten

10.6.6.10. Subtest 6.10 "back to back loading" des TODR

Es wird abgeprüft, ob das TODR auf ein unmittelbar hintereinander folgendes Laden mit unterschiedlichen bzw. gleichen Werten richtig reagiert.

Prüfschritte:

1. Laden 1. Wert (55555555 hex) in TODR
2. unmittelbares Laden 2. Wert (AAAAAAAA hex)
3. Programmverzögerung fuer 20 ms
4. Lesen TODR und Vergleichen mit 2. Wert

10.6.6.11. Subtest 6.11 Schreiben/Lesen TODR

Es wird abgeprüft, ob unmittelbar nach Einschreiben von -1 in das TODR dieser Wert auch wieder richtig ausgelesen werden kann, ansonsten erfolgt Fehlerausschrift.

10.6.6.12. Subtest 6.12 Zählfunktion des TODR

Es wird die Zählfolge des TODR abgetestet, indem in Zeitintervallen von etwa 14 bis 15 ms ein C-Bit erwartet wird.

Prüfschritte:

1. Setzen allg. Register mit Zählwert bzw. als C-Bitzähler
2. Laden des Intervalltimer mit einem Wert größer als der Erwartungsintervallzeitwert (timeout ca.20 ms)
3. Laden TODR mit einem Wert zur Erzeugung d. C-Bit
4. Intervalltimerinterrupt zulassen
5. Warten auf Interrupt durch TODR
6. Wenn Interrupt innerhalb timeout, dann C-Bit- Zähler erhöhen und Sprung zu 2., ansonsten Fehlerausschrift und Übergang zu nächstem Subtest

10.6.7. Test 7 -Speicherverwaltung MM (memory management)

Testinitialisierung:

1. Setzen der Basis des virtuellen Adreßspeicherraumes (BVAS)
2. zunächst werden 6 Seiten des Puffers des Supervisors genommen, jeweils 2 Seiten (P1 und P0) von jedem der 3 Bereiche
3. Jeder der 3 Bereiche, beginnend bei Langwort 3 wird in den BVAS geladen

4. Das Längenregister jeder der 3 Bereiche wird in Speicherzellen geladen, deren Adressen in den BVAS, beginnend bei Langwort 6, abgespeichert werden.

Der Test besteht aus einer Schleife, die jeweils folgende Teilroutinen umfaßt:

1. Aufstellen einer Tabelle, die 6 Testparameter enthält:
   - Zugriffstyp (access type)
   - Adressraum (address space)
   - V-Bit
   - Speicherschutz (protection)
   - Zugriffsgröße (access size)
   - Bytezuordnung (byte alignment, position)
2. Simulierter Zugriff, um festzustellen, welche Exceptions dabei auftreten
3. Versuch, über die Zugriffsrechte hinauszugehen und Prüfung der Reaktion hierauf
4. Löschen von Adressen im Adressübersetzungspuffer TB (translation buffer), die in einer bezüglich des Programms internen Statustabelle spezifiziert sind, wobei die Zuordnung über einen Befehl erfolgt,dessen Typ in der Tabelle spezifiziert ist.
5. Aufstellen der nächsten Tabelle, bis alle Bereiche genügend abgetestet sind.
6. Wiederholung der Testschleife

Bei erkannten Fehlern erfolgen Ausschriften, die die aktuellen Testparameter enthalten und den Fehler verbal beschreiben.

10.6.8. Test 8 - Fehlerhafter Kernel-Stack

Dieser Test überprüft die Funktion der Rechnerarchitektur wenn der Kernel-Stack-Pointer eine ungültige Adresse enthält oder auf einen zugriffsgeschützten Speicherbereich zeigt.

10.6.8.1. Subtest 8.1 Zugriffsschutz bei ungültigem Kernel-Stack

Dieser Subtest prüft die Reaktion bei Befehlen, die den Kernel-Stack benutzen und eine Zugriffsverletzung zur Speicherverwaltung (memory management access violation) bewirken, wenn der Kernel-Stack gegen Zugriff geschützt ist.

Prüfschritte:

1. Einschalten der Speicherverwaltung (MM)
2. Ausführung eines Befehls, der zur Zugriffsverletzung führt
3. Erwarten einer Exception
   3.1 wenn nicht, Ausschalten MM, Fehlerausschrift, Ende
   3.2 wenn ja, Ausschalten MM, Prüfen des Interruptstacks und der darin enthaltenen Parameter, wenn i.O. dann zu 1. , bis alle derartigen Befehle abgeprüft sind, wenn nicht i.O. dann Fehlerausschrift, Ende
4. Ende, Übergang zum nächsten Subtest

10.6.8.2. Subtest 8.2 Befehl BPT zum KSP (kernel stack pointer)

In diesem Subtest wird abgeprüft, ob ein BPT-Befehl zu einem Programmabbruch wegen ungültigem Kernel-Stack (kernel stack not valid abort) führt, wenn der Kernel-Stack zugriffsgeschützt ist.

Prüfschritte:

1. Laden des Vektors "Zugriffsverletzung" (Vektor 20, access control violation)
2. Laden des Vektors "ungültiger Kernel-Stack" (Vektor 08, kernel stack not valid)
3. Einschalten Speicherverwaltung (MM) und damit Speicherschutz
4. Ausführung eines BPT-Befehls (break point fault)
5. Erwarten einer Exception, ansonsten Fehlerausschrift
6. Prüfen des Interruptstack und der darin enthaltenen Informationen, ggf. Fehlerausschrift
7. Laden des Vektors "Intervallzeitgeber" (Vektor C0, interval clock, IPL=18 mit Vektorkode = 1 (Interrupt))
8. wie 1.
9. wie 2.
10. wie 3. und Starten des Zeitgebers
11. Warteschleife für Interrupt (timeout)
12. Ausschalten MM, Prüfung des Ergebnisses wie bei 6. ggf. Fehlerausschrift, Rückspeichern des KSP

10.6.9. Test 9 - Modusänderung zu schreibgeschütztem Stack

Es wird abgetestet, ob eine Zugriffsverletzungs-Exception auftritt, wenn Modusänderungsbefehle (CHMx) zu einem ungültigem Destinationstack ausgeführt werden.

Prüfschritte:

1. Retten der verwendeten Vektoren
2. Laden des Vektors "Zugriffsverletzung" (Vektor 20) mit [01:00] = 1 (Interruptstack)
3. Definieren eines Puffers für den Stack
4. diesen Puffer schreibschützen
5. folgende Schritte jeweils für alle Modi K (kernel), E (executive), S (supervisor) und U (user) :
   - 5.1 Laden des Interruptstack
   - 5.2 Laden der CHMx-Vektoren (40, 44, 48 und 4C)
   - 5.3 Zeiger für Modusstack auf ungültige Seite setzen
   - 5.4 REI zum User-Modus
   - 5.5 Ausführung des betreffenden CHMx-Befehls
   - 5.6 Fehlerausschrift wenn keine Exception auftritt,ansonsten 5.7
   - 5.7 Prüfen Prozeßstatus und Rückspeichern des Modusstackpointers, Übergang in den Ausgangsmodus und Rückspeichern aller Vektoren

10.6.10. Test 10 - Test der Befehle PROBEx

Durch die PROBEx-Befehle ist es möglich, den Lese- (PROBER) bzw. den Schreibzugriffsschutz (PROBEW) von Speicherseiten (virtuelle Adressen) abzufragen. Dieser Test prüft die richtige Ausführung dieser Befehle, wobei die Argumente jeweils variiert werden.

10.6.10.1. Subtest 10.1
Test des Befehls PROBER

In diesem Subtest wird der Befehl PROBER für spezifische Kombinationen von Speicherregionen, priviligierten Ebenen (levels), Speicherseitenschutz und PSW-NZVC-Verbindungen überprüft. Die Prüfung erfolgt durch Vergleich erwarteter PSW-Flags (N, Z, V und C) mit den nach Abarbeitung des PROBER erhaltenen.
Folgende Modifikationen werden abgetestet:

1. Speicherregionen: System, P1-Raum, P0-Raum
2. Modusargument: PSL <PRV MODE> = Kernel, Executive, Supervisor und User
3. Seitenschutz: Ebenen 00 bis 15
4. PSW-NZVC-Vorbedingungen: "alles 0" und "alles 1"

10.6.10.2. Subtest 10.2
Test des Befehls PROBEW

Dieser Subtest läuft nach dem gleichen Algorithmus und mit den gleichen Modifikationen ab, wie der Subtest 10.1 nur bezogen auf den Befehl PROBEW.

10.6.11. Test 11 - Test der Befehle SVPCTX und LDPCTX

Dieser Test prüft in den ersten 4 Subtests die richtige Ausführung des Befehls SVPCTX (save process context) und in den Subtests 11.5 bis 11.9 den Befehl LDPCTX (load process context). Dieser Test kann nur im stand-alone-Modus, d.h. durch direktes Booten des Supervisors und nicht vom Betriebssystem aus gestartet, abgearbeitet werden. Der Versuch, ihn im User-Modus zu starten führt zum Programmabbruch.

10.6.11.1. Subtest 11.1
SVPCTX mit normalen Daten,
Interruptstack

In diesem Subtest wird der Befehl SVPCTX im Kernel-Modus, dem einzigen legalen Modus für diesen Befehl, abgearbeitet. Die Ausführung wird vom Interruptstack aus gestartet und auch dort beendet. Ist die Befehlsabarbeitung fehlerhaft, so ist zu erwarten, daß auch der Stack verfälscht ist, wodurch ein weiteres Testen sinnlos wird. Nach Befehlsausführung erfolgt ein Soll-/Ist-Vergleich der Datenfelder. Bei Nichtübereinstimmung erfolgt Fehlerausschrift und Übergang zum nächsten Subtest.

Mögliche Fehler:

- falscher Stack
- unerwartete Exceptions

10.6.11.2. Subtest 11.2
SVPCTX mit normalen Daten,
Kernelstack

Prinzipiell gleiche Abarbeitung wie Subtest 11.1 nur bezüglich Kernel-Stack

10.6.11.3. Subtest 11.3
SVPCTX mit extremen Daten
("alles 0")

Prinzipiell gleiche Abarbeitung wie Subtest 11.1 , nur daß zuvor alle Register R0 bis FP außer KSP- und MM-Register gelöscht werden ("alles 0").

10.6.11.4. Subtest 11.4
SVPCTX mit extremen Daten
("alles 1")

Prinzipiell gleiche Abarbeitung wie Subtest 11.3 , nur daß die genannten Register auf "alles 1" gesetzt werden.

10.6.11.5. Subtest 11.5
LDPCTX mit normalen Daten,
Kernel-Stack

Prinzipiell gleiche Abarbeitung wie Subtest 11.2 , nur bezüglich des Befehls LDPCTX

10.6.11.6. Subtest 11.6
LDPCTX mit normalen Daten
vom Interrupt-Stack zum
Kernel-Stack

Prinzipiell gleiche Abarbeitung wie Subtest 11.5 , nur Transfer vom Interrupt-Stack zum Kernel-Stack

10.6.11.7. Subtest 11.7
LDPCTX mit reservierten Operanden

In diesem Subtest wird versucht, das Feld 1 im Kernel-Stack, das "0" sein muß mit "1" zu laden.

Fehlermöglichkeiten:

- Zerstörung des Kernel-Stack
- unerwartete Exceptions
- keine erwartete Exception

10.6.11.8. Subtest 11.8
LDPCTX normal im AST-Level

Prinzipiell gleiche Abarbeitung wie Subtest 11.5, jedoch AST-Level auf "0" gesetzt.

Fehlermöglichkeiten:

- kein Umschalten in den Kernel-Stack
- unerwartete Exceptions
- Datenfehler

10.6.11.9. Subtest 11.9
wie Subtest 11.7 mit "0"-Feld 3

10.7. Abkürzungsverzeichnis

Alle Abkürzungen , die Befehle, Signale oder Funktionsgruppen betreffen, sind dem Anhang des Diagnosehandbuches Rechner zu entnehmen. Innerhalb dieser Programmbeschreibung werden darüber hinaus folgende Abkürzungen verwendet:

DS	Diagnose-Supervisor
FPS	Festplattenspeicher
FSE	Folienspeichereinheit
MM	Speicherverwaltung (memory-management)
CPU	Zentrale Verarbeitungseinheit

10.8. Auflistung der Tests

Das Gesamttestprogramm wird in folgende Einzeltests unterteilt:

Test Nr.	1:	Interrupt-Test
	2:	Exception-Test
	3:	REI-Befehlstest
	4:	Test der Change-Mode-Befehle
	5:	Test der Prozessorregister-Befehle
	6:	Test des Intervallzeitgebers
	7:	Test der Speicherverwaltung (MM)
	8:	Test "Systemstack nicht gültig"
	9:	CHMx-Befehle zu schreibgeschützten Stacks
	10:	Test der PROBE-Befehle
	11:	Test der Prozeßcontext-Befehle

11. PVKAM: Speichertest-Programm

11.1.

Das Diagnoseprogramm PVKAM (Dok.-Nr. 1.57.700007.5/67) ist ein Programm der Ebene 2R (User-Mode). Es ist bestimmt für den Test des Hauptspeichers K 3581 (MSA40) des Rechnersystems Robotron K 1840. Beim Start wird nacheinander die Funktionsfähigkeit der angeschlossenen Speicher getestet.
Die Fehlerauflösung erfolgt bis auf den fehlerhaften Modul. Der Speichertest PVKAM ist das einzige online-Speicher-Diagnoseprogramm, das für das RVS K 1840 verfügbar ist.

11.2. Testbedingungen

11.2.1. Hardware

Das Programm erfordert:
die CPU K 2810 (KAA40)
ein Terminal, angeschlossen an den Prozessor

11.2.2. Software

Für die Abarbeitung des Programms ist erforderlich:
- ein laufendes SVP 1800
- Diagnose-Supervisor
- Zugriff zu dem Verzeichnis von Privilegien und Quotas
  Zusätzlich zu den Standard-Privilegien (s. Supervisor-Beschreibung) werden die Privilegien PFNMAP und CMKRNL benötigt.

11.3. Voraussetzungen

Die notwendige Hardware für den Bootstrap-Prozeß des SVP 1800-Operationssystems ist für eine normale Funktion Voraussetzung.

11.4. Bedienungshinweise

Es ist kein ATTACH- oder SELECT-Kommando erforderlich.

Zum Start des Programms:
- Start des Diagnose-Supervisors
- Laden des Diagnoseprogramms
- Eingabe des Start-Kommandos

11.5. Funktionsbeschreibung

11.5.1. Übersicht

Das Programm besteht aus zwei Tests; Test 1 ist der einzige beim Standard laufende Test. Der zweite Test wird im Detail später beschrieben. Das Programm ist in BLISS 18 und MACRO 18 geschrieben.

11.5.2. Programmgröße

Das Programm umfaßt annähernd 26 Kbyte.

11.5.3. Programmlaufzeit

Test 1 beansprucht ca. 10 s pro Mbyte getesteten Speicher. Die Laufzeit des Test 2 beträgt ca. 1 min/0,5 Mbyte, wenn kein weiterer Prozeß läuft. Sie ist davon abhängig, wieviel Speicherraum, nach fallenden Adressen hin, zusammenhängend in dem Prozeß-Working-Set verfügbar ist. Die Laufzeit ist damit abhängig vom geladenen System. Die oben angegebenen Zeiten beziehen sich auf ein System, welches unter anderem interaktive oder Stapel-Nutzung ermöglicht. Weitere Prozesse konkurrieren mit dem Diagnoseprogramm um die CPU-Laufzeit und beeinflussen folglich die Programmlaufzeit.

11.5.4. Abarbeitungsbedingungen

- Der Cache-Speicher ist gesperrt, wenn der Test 1 läuft; die Nutzer dieses Programms sollten wissen, daß bei der Systemausführung Kollisionen möglich sind und sollten dies vor der Programmausführung berücksichtigen.
- Die in der SVP-Fehlerprotokollierung enthaltene Latenz ist eine große Unterstützung, damit bei vielen Wiederholungen des Programmlaufes alle ermittelten Fehler in der Reihenfolge ihrer Entdeckung erfaßt werden.
- Wenn die Methode der gesicherten Fehlermitteilung während des Laufs des Dienstprogramms SYE genutzt wird, besteht das Risiko, daß das SVP 1800 eine neue Fehlerprotokoll-Datei starten will, während der ERRFMT-Prozeß versucht, eine Eintragung vorzunehmen und SYE läuft.
- Extern überlappte Speicher sind doppelt zu testen.
- Alle existierenden Speicher sind zu testen, mit einer Ausnahme:
  der lokale Speicher der eingegliederten (sekundären) CPU in dem RVS K 1840 kann nicht getestet werden.

11.5.5. Flag-Verwendung

Ereignis-Flag 1 - Freigabe der Mitteilung der korrigierbaren Lesedaten-Fehler (1-bit-Fehler)
Ereignis-Flag 10 - Freigabe der Mitteilung der Speicher-Konfiguration
Ereignis-Flag 11 - Freigabe der Operator-Mitteilung in Mbyte-Bereichen (Test 1)
Ereignis-Flag 12 - Freigabe der Operator-Mitteilung über jedes Testmuster

Weitere Informationen zu den Ereignis-Flags 11 und 12 siehe Pkt. 1.5.8. "Programmitteilungen".

11.5.6. Allgemeine Programmanforderungen

keine

11.5.7. Fehlermitteilungen

Das Programm ist dafür bestimmt, korrigierbare Lesedaten-Fehler (1-bit-Fehler, die erkannt werden und korrigiert werden können) oder Fehler substituierter Lesedaten (2- oder Mehr-bit-Fehler, die erkannt, aber nicht korrigiert werden können) in den unterstützten Speichern zu ermitteln. Ausserdem werden auftretende unbekannte Fehler ausgewiesen. Die primäre Fehlererkennung ist in der Überwachung des System-Fehlerprotokoll-Bereiches angeordnet.
Bei aufgetretenen Fehlern durchläuft das SYE-Dienstprogramm den Cleanup-Code.
Das Programm ist zur Ermittlung und Mitteilung aller Speicherfehler-Bedingungen und bleibenden Fehlern bestimmt.

11.5.8. Programmitteilungen

Das Ereignis-Flag 11 bewirkt, wenn durch den Operator gesetzt, eine Mitteilung, wenn das Programm bei der Ausführung des Tests 1 einen Mbyte-Bereich getestet hat.
Das Ereignis-Flag 12 bewirkt, wenn durch den Operator gesetzt, eine Mitteilung während der Ausführung des Tests 2, wenn das Programm ein Testmuster beginnt.

11.5.9. Leistungsparameter des Testobjekts

Die Zahl der ermittelten korrigierbaren Lesedaten (CRD), substituierten Lesedaten (RDS) und unbekannten Fehler während des Programmlaufes sind in einem Fehlerprotokoll enthalten. Dies stellt einen Auszug aus der Endemeldung eines Laufes oder des SUMMARY-Kommandos dar.

11.5.10. Aktionen bei unerwarteten Fehlern

Das Programm ist konzipiert für die Mitteilung und Wiederherstellung aus allen Hardware-Fehlerumständen, ausgenommen Netzfehler.

11.5.11. Testfolge bei mehreren Einheiten

Alle Speicher werden bei jedem Lauf des Programms getestet (einzeln).

11.5.12. Einstellungen für Ablauf unter AFT

nicht angewendet

11.5.13. Testsektionen

Das Programm hat drei Testsektionen:

DEFAULT	(MEMFT)	(Standard) -	Lauf des Tests 1
CONFT		-	" " " 2
ALL	(MEMFT+CONFT)	-	" " " 1+2

11.5.14. Verschiedenes

entfällt

11.6. Testbeschreibungen

11.6.1. Test 1 (MEMFT)

Dieser Test stellt die MEMFT-Sektion in dem System-Funktionstest dar. Der Test 1 sperrt den Cache, liest von allen existierenden physischen Speichern einzeln ein Langwort und gibt den Cache wieder frei. Ein AST wird an den Prozeß gesendet, wenn ein Fehler ermittelt wurde; die Fehlermitteilung wird in der AST-Routine ausgeführt. Wenn in einem Speicher Fehler ermittelt wurden, gibt das Programm eine Mitteilung mit Angabe des Fehlers und des fehlerhaften Moduls.

11.6.2. Test 2 (CONFT)

Dieser Test stellt die CONFT-Sektion in dem System-Funktionstest dar. Der Test 2 beginnt mit der Verriegelung einer großen Anzahl Pages im Speicher, die vom Operationssystem freigegeben sind. Es werden nur die freien Pages des Working-Sets des eigenen Prozesses getestet. Danach wird der gesamte Bereich beschrieben, gelesen und geprüft, mit 7 Datenmustern. Zuletzt wird dieser abgegrenzte Bereich wieder freigegeben.
Für den Speichertest werden folgende Testmuster verwendet (Angaben hex.)

Muster 1: 00000000
Muster 2: FFFFFFFF
Muster 3: 55555555
Muster 4: 33333333
Muster 5: 0F0F0F0F
Muster 6: 00FF00FF
Muster 7: 0000FFFF

Ein AST wird zum Prozeß ausgelöst, wenn ein Fehler erkannt wurde; die Fehlermitteilung wird in der AST-Routine ausgeführt. Da erwartet werden muß, daß außer dem Diagnose-Prozeß andere Prozesse während des Laufs des Test 2 (CONFT im System-Funktionstest) anwesend sind, ist der Cache-Speicher zur Vermeidung von Konflikten beim Programmablauf während dieses Tests nicht gesperrt. Wenn im Speicher ein Fehler ermittelt ist, teilt das Programm den Fehler und die fehlerhafte Einheit mit.

11.7. Abkürzungsverzeichnis

AST - asynchroner System-Trap
CRD - korrigierbare Lesedaten (correctable read data)
RDS - substituierte Lesedaten (read data substitute)

11.8. Auflistung der Tests

Test 1 : Speicherlesen
Test 2 : Speicherschreiben/-lesen und Vergleichen

12. PVSBA - K 1840 Autosizer-Test

12.1. Zusammenfassung

PVSBA (Dok.-Nr. 1.57.700008.3/67) ist kein Programm, sondern ein Hilfsmittel. PVSBA läuft innerhalb der Ebene 3 (Stand-alone). Es ermittelt die aktuelle Konfiguration eines RVS K 1840 im Stand-alone-Modus unter Steuerung des Diagnose-Supervisor. Das Programm generiert die ATTACH-Kommandos für alle Geräte der jeweiligen Konfiguration des RVS K 1840. Bei entsprechender Testauswahl können die generierten ATTACH-Folgen manuell bearbeitet werden. Diese Datei kann dann als Konfigurationskommandodatei zur Abarbeitung der Diagnoseprogramme (auch im User-Modus) genutzt werden.

12.2. Testbedingungen

12.2.1. Hardware

- Ausstattung:
  RVS K 1840
  beliebige Gerätekonfiguration

12.2.2. Software

Diagnose-Supervisor, über BOOT-Kommando gestartet

12.3. Voraussetzungen

Der Test setzt voraus, daß bei einer Gerätekonfiguration mit den Magnetbandgeräten CM 530x die Bänder geladen und die Geräte online gesetzt sind. Andernfalls wird für diese keine ATTACH-Folge generiert. Gerät 0 wird jedoch immer mitgeneriert, vorausgesetzt, daß die Geräteregister des Kontrollers erkannt werden.

12.4. Bedienungshinweise

Nachdem der Diagnose-Supervisor über BOOT-Kommando geladen und gestartet wurde, kann PVSBA sofort abgearbeitet werden. Da PVSBA die ATTACH-Folgen selbst generiert, entfällt natürlich das übliche Anschließen der Geräte durch ATTACH-Kommandos. Werden dennoch ATTACH-Kommandos vorher ausgeführt, so werden für bereits angeschlossene Geräte die Parameter evtl. überschrieben und weitere ATTACH-Folgen für die restlichen Geräte ergänzt.

12.5. Funktionsbeschreibung

12.5.1. Übersicht

PVSBA läuft unter dem Diagnose-Supervisor und paßt sich den RVS-Diagnosebedingungen an. Es hat einen Kopfmodul, welcher Initialisierungs-, CLEAN-UP und eine Reihe anderer Subroutinen enthält. Außerdem sind hier alle globalen Text-, Daten- und Puffervereinbarungen enthalten. Die Programmablauffolgen für Test 1-3 sind ebenfalls im Kopfmodul enthalten.
Modul 1 enthält wichtige Macrodefinitionen sowie Konfigurationstabellen für die einzelnen Adapter mit den zugehörigen Geräten. Außerdem sind Subroutinen zur Austestung der verschiedenen Adaptertypen enthalten.
Modul 2 enthält wiederum Macrodefinitionen, desweiteren Fehlertextvereinbarungen und Subroutinen zum Aufbau der ATTACH-Tabelle.
Modul 3 enthält ebenfalls einige Macrodefinitionen und die Datenbasen für die Geräte-Deskriptoren.
Modul 4 enthält neben Macrodefinitionen und Fehlertextvereinbarungen vor allem die Dateibearbeitung und die Transferprogramme zur Konsoldiskette.
Die Module 5-7 enthalten die Gerätedriver (z.B. den Floppy-Disk-Driver).

12.5.2. Programmgröße

Speicherbelegung von 200(x)...BFFF(x), das sind BE00(x), d.h. 48640 byte.

12.5.3. Programmlaufzeit

Insgesamt ca. 5 sec für mittlere Gerätekonfiguration

12.5.4. Abarbeitungsbedingungen

Der Diagnose-Supervisor beginnt den Programmablauf immer mit der Initialisierungsroutine und schließt es mit der CLEAN-UP-Routine ab.

- Initialisierungsroutine
  Die Routine testet aus, ob der Diagnose-Supervisor im Stand-alone-Modus arbeitet. Wenn nicht, erfolgt eine entsprechende Fehlerausschrift.

- CLEAN-UP-Routine
  In dieser Routine wird nur die Speicherverwaltung in den entsprechenden

internen CPU-Registern zurückgesetzt.

12.5.5. Flagverwendung

Das Programm verwendet nur das QUICK-Flag. Dieses, um zu entscheiden, ob die Prüfung für die Terminals an den AZ16 durchzuführen ist. Das Flag ist ein Supervisor-Flag und ist durch die entsprechenden Kommandos des Supervisor zu setzen bzw. zu löschen.
Wenn das Flag gesetzt ist, werden die Terminals der AZ16 nicht geprüft. Das ist anzuwenden, wenn keine Terminals vorhanden sind und eine rasche Konfiguration benötigt wird.
Mit gesetztem QUICK-Flag geht die Programmverbindung sehr schnell voran. Während PVSBA arbeitet, werden keine Mitteilungen ausgegeben.

12.5.6. Allgemeine Programmanforderungen

entfällt

12.5.7. Fehlermitteilung

Die Fehlermitteilungen erfolgen verbal und sind in sich erklärend (siehe auch Beispiele in den Testbeschreibungen).

Beispiel: PVSBA wird im User-Modus gestartet

```
DS> ST
.. Program: PVSBA -AUTOSIZER LEVEL 3, revision 1.0, 3 tests, at 10:12:31.68.
! *** THIS PROGRAM IS NOT SUPPOSED TO BE RUN UNDER SVP ***

.. Aborted program at pass 0, Initialization section, PC 00005C30.
DS>
```

12.5.8. Programmitteilungen

entfällt

12.5.9. Leistungsparameter des Testobjektes

ATTACH-Folgen für die angeschlossenen Geräte an der AMF18 werden nicht generiert.

12.5.10. Aktionen bei unerwarteten Fehlern

entfällt

12.5.11. Testfolge bei mehreren Einheiten

entfällt

12.5.12. Einstellungen für Ablauf unter AFT

entfällt

12.5.13. Testsektionen

PVSBA besteht aus drei Testsektionen.

```
DS> SH SEC
PVSBA - AUTOSIZER LEVEL 3 contains the following sections:
DEFAULT, SELFTEST, MANUAL
DS>
```

1) DEFAULT - Standardsektion
2) SELFTEST - Testsektion mit Eigentest
   Test 2
3) MANUAL - Testsektion mit manuellen Eingriffen
   Test 3

12.6. Testbeschreibungen

PVSBA enthält 3 Tests.

12.6.1. Test 01: DEFAULT MODE Standard-Modus

In diesem Test des Programmes sind mehrere Operationen zusammengefaßt. Zunächst werden die angeschlossenen Adapter und Geräte durch spezielle Testfolgen ermittelt und daraus entsprechende ATTACH-Folgen generiert. Anschließend werden alle ATTACH-Kommandos dem Supervisor zur Ausführung übergeben.
Um zu zeigen, daß das Programm arbeitet, wird die Ausschrift "PVSBA working" erzeugt, falls AZ16 verfügbar sind. PVSBA ergänzt die Liste der angeschlossenen Geräte bzw. überschreibt die Parameter für bereits existierende Gerätenamen.
Das nachfolgende Beispiel zeigt die Liste der angeschlossenen Geräte vor und nach dem Start von PVSBA.

```
DS> SH DEV
_CSA0   console  HUB    00000000
_RH1    RHA40    HUB    60012000  TR=9. BR=5.
_DRA0   PM05     _RH1   60012400
DS> ST

.. Program: PVSBA - AUTOSIZER LEVEL 3, revision 1.0, 3 tests, at 10:23:18.51.

.. End of run, 0 errors detected, pass count is 1,
   time is 7-JAN-1987 10:23:20.24
```

```
DS> SH DEV
_CSA0    console  HUB    00000000
_RH1     RHA40    HUB    60012000  TR=9. BR=5.
_DRA0    PM05     _RH1   60012400
_KA0     KAA40    HUB    00000000  G-floating instructions=Yes H-floating instruc
tions=Yes WCS last address=00001FFF(X) Accelerator type=1.
_MS0     MSA40    HUB    60002000  TR=1. Number of arrays=1.
_DW0     DWA40    HUB    60006000  TR=3. BR=4.
_XEA0    SNA16    _DW0   6013F948  CSR=774510(O) VECTOR=000120(O) BR=5.
_MEA0    BM16     _DW0   6013F550  CSR=772520(O) VECTOR=000224(O) BR=5.
_XGA0    AMF18S   _DW0   6013E0E0  CSR=760340(O) Vector=000314(O) BR=5. External
Wrap=NONE
_TXA     AMF18A   _DW0   6013E0E0  CSR=760340(O) Vector=000314(O) BR=5. Active Li
nes=000377(O) Baud Rate=9600 Loopback Type=INTERNAL SKRBUS Init Jumper=Yes
_LCA     AMF18P   _DW0   6013E0E0  CSR=760340(O) Vector=000314(O) BR=5. Port Devi
ce=LP
_RH0     RHA40    HUB    60010000  TR=8. BR=5.
_DBA4    PP06     _RH0   60010600
_DBA5    PP05     _RH0   60010680
_DRB0    PM05     _RH1   60012400
```

12.6.2. Test 02: SELFTEST MODE Eigentest-Modus

Der Test wird vom Operator im Dialogverkehr gesteuert. Der Testablauf beginnt mit einer Aufforderung zur Kommandoeingabe. Folgende Kommandos stehen zur Verfügung:

ATTACH, CHANGE, EXIT, HELP, LIST, READ, SIZE, WRITE

12.6.2.1. ATTACH

Das Kommando ist nur nach Ausführung eines SIZE- oder READ-Kommandos gültig. Für jedes Gerät werden in einem Datenpuffer die generierten Informationen dem Diagnose-Supervisor übergeben. Alle ATTACH-Folgen, die einen gemeinsamen LINK- bzw. GENERIC-Feldwert haben, sind dem Supervisor bereits bekannt und werden nur aktualisiert. Alle anderen ATTACH-Folgen werden hinzugefügt.

Format:

ATTACH

12.6.2.2. CHANGE

Das Kommando ist nur nach Ausführung eines SIZE- oder READ-Kommandos gültig. Es dient der manuellen Bearbeitung der einzelnen Parameter einer bereits generierten ATTACH-Folge für ein Gerät. Diese Parameter sind entsprechend ihrer Bedeutung in Felder eingeordnet. Die Bearbeitung erfolgt in der Reihenfolge:

- Geräteauswahl (GENERIC-Name)
- Feldauswahl (TYPE, LINK, GENERIC, CSR, VECTOR, BR)
- Wertangabe für das spezifische Feld

Wenn Gerät, Feld oder Wert nicht spezifiziert sind, wird eine gültige Eingabe des Gerätenamens einer Feldvereinbarung bzw. des Wertes vom Operator gefordert. Wenn ein Feld nicht spezifiziert ist, wird der Operator aufgefordert, diesen einzugeben.
Für jede Eingabe werden nach der Operatoraufforderung zunächst die möglichen Antworten für diese Eingabe ausgegeben. Für die Geräteanforderung werden die Namen aller bekannten Geräte ausgegeben (durch SIZE/READ-Kommando generiert). Für die Feldanforderung werden alle gültigen Feldnamen ausgewiesen.
Für die Wertanforderung werden Format und Grenzen angegeben. Die Felder müssen in der Anweisung des ATTACH-Kommandos spezifiziert sein. Nur die Felder müssen gewechselt werden, die im Kommando erscheinen. Alle Felder für ein Gerät sind überprüft, wenn eines von ihnen gewechselt wird.
Falsche oder ungültige Werte führen zu einer Operatoranforderung für den richtigen Wert. Gemeinsame Felder für alle Geräte sind TYPE, LINK und GENERIC.
Wird bei einer Anforderung nachfolgend auf Space oder Tab ein Carriage Return eingegeben, kehrt PVSBA zum Kommandomodus zurück, ohne die CHANGE-Anweisung auszuführen.

Format:

```
CHANGE [<device> [<field> [=<value>] ...]]
```

Das nachfolgende Beispiel zeigt anhand der Bearbeitung von "MEA0" anschaulich die prinzipielle Wirkung des CHANGE-Kommandos.
Vor dem CHANGE-Kommando muß das SIZE-Kommando gegeben werden. Im Beispiel wurde im Anschluß daran noch das LIST-Kommando gegeben, um das Ergebnis der Bearbeitung durch das CHANGE-Kommando zu verdeutlichen. Mit dem EXIT-Kommando wird der Test beendet.

```
DS> ST/SE:SE
.. Program: PVSBA - AUTOSIZER LEVEL 3, revision 1.0, 3 tests,
   at 10:46:11.51.

Test 2: SELFTEST MODE
COMMAND? SI
! AUTOMATIC SIZING PROGRAM.
! THIS IS A TOOL AND NOT A DIAGNOSTIC PROGRAM.
! CONFIGURATION FILE FOR SYSTEM.
! COMPUTER GENERATED FOR PROCESSOR SERIAL # 3070 CPU TYPE 1
! NUMERIC VALUES WITH A LEADING ZERO ARE ASSIGNED.
! FILE VALID ONLY FOR STANDARD HARDWARE CONFIGURATION.
! TIME IS   7-JAN-1987 10:46:19.91
!
! DEFINE PROCESSOR...
!
DS> ATTACH KAA40 HUB KA0 YES YES 1FFF 1
!
! DEFINE MEMORY...
!
DS> ATTACH MSA40 HUB MS0 1  1
!
! DEFINE SKRBUS ADAPTERS...
!
DS> ATTACH DWA40 HUB DW0 3 4            ! SKRBUS I/O SPACE NUMBER  0
!
DS> ATTACH SNA16 DW0 XEA0 774510 0120 05
DS> ATTACH BM16 DW0 MEA0 772520 0224 05
DS> ATTACH BM16 DW0 MEA1 772520 0224 05
```

```
DS> ATTACH AMF18S DW0 XGA0 760340 0314 05 NONE
DS> ATTACH AMF18A DW0 TXA 760340 0314 05 0377 9600 INTERNAL YES
DS> ATTACH AMF18P DW0 LCA 760340 0314 05 LP
!
! DEFINE MSBUS   ADAPTERS...
!
DS> ATTACH RHA40 HUB RH0 8 5
!
DS> ATTACH PP06 RHO DBA4
DS> ATTACH PP05 RH0 DBA5
!
DS> ATTACH RHA40 HUB RH1 9 5
!
DS> ATTACH PM05 RH1 DRB0
COMMAND? CH
VALID RESPONSES ARE:
KA0, MS0, DW0, XEA0, MEA0, MEA1,
XGA0, TXA, LCA, RH0, DBA4, DBA5,
RH1, DRB0
ENTER DEVICE: MEA0
VALID RESPONSES ARE:
TYPE, LINK, GENERIC, CSR, VECTOR, BR
ENTER FIELD: TYP
INVALID FIELD NAME.  'TYP'
ENTER FIELD: TYPE
VALID RESPONSES ARE:
AMF18, AMF18A, AMF18P, AMF18S, AR16W,
DWA40, AZ16, KAA40, DN13, DT52, DP23,
DP27, MSA40, RHA40, PM80, PP05, PP06,
BM16, SNA16, GP11, GD04, TG911, TG918
ENTER VALUE: BU89\9\0\08UB\BS16
COMMAND? CH
VALID RESPONSES ARE:
KA0, MS0, DW0, XEA0, MEA0, MEA1,
XGA0, TXA, LCA, RH0, DBA4, DBA5,
RH1, DRB0
ENTER DEVICE: MEA0
VALID RESPONSES ARE:
TYPE, LINK, GENERIC, CSR, VECTOR, BR
ENTER FIELD: LINK
VALID RESPONSES ARE:
HUB, KA0, MS0, DW0, XEA0, MEA1,
XGA0, TXA, LCA, RHO, DBA4, DBA5,
RH1, DRB0
ENTER VALUE: HUB
COMMAND? CH
VALID RESPONSES ARE:
MEA0, KA0, MS0, DW0, XEA0, MEA1,
XGA0, TXA, LCA, RHO, DBA4, DBA5,
RH1, DRB0
ENTER DEVICE: MEA0
VALID RESPONSES ARE:
TYPE, LINK, GENERIC, CSR, VECTOR, BR
ENTER FIELD: GENERIC
ENTER ALPHANUMERIC VALUE.
ENTER VALUE: MT03
COMMAND? CH
VALID RESPONSES ARE:
```

```
MT03, KA0, MS0, DW0, XEA0, MEA1,
XGA0, TXA, LCA, RHO, DBA4, DBA5,
RH1, DRB0
ENTER DEVICE: MT03
VALID RESPONSES ARE:
TYPE, LINK, GENERIC, CSR, VECTOR, BR
ENTER FIELD: CSR
ENTER VALUE: OCTAL 760000 THRU 777776:  122
INVALID VALUE.  '122'
ENTER VALUE: OCTAL 760000 THRU 777776:  000122
INVALID VALUE.  '000122'
ENTER VALUE: OCTAL 760000 THRU 777776:  767773
COMMAND? CH
VALID RESPONSES ARE:
MT03, KA0, MS0, DW0, XEA0, MEA1,
XGA0, TXA, LCA, RHO, DBA4, DBA5,
RH1, DRB0
ENTER DEVICE: MT03
VALID RESPONSES ARE:
TYPE, LINK, GENERIC, CSR, VECTOR, BR
ENTER FIELD: VECTOR
ENTER VALUE: OCTAL 000002 THRU 000776: 774\4\3
COMMAND? CH
VALID RESPONSES ARE:
MT03, KA0, MS0, DW0, XEA0, MEA1,
XGA0, TXA, LCA, RHO, DBA4, DBA5,
RH1, DRB0
ENTER DEVICE: MT03
VALID RESPONSES ARE:
TYPE, LINK, GENERIC, CSR, VECTOR, BR
ENTER FIELD: BR
ENTER VALUE: DECIMAL 4 THRU 7:  8
INVALID VALUE.  '8'
ENTER VALUE: DECIMAL 4 THRU 7:  7
COMMAND? CH
VALID RESPONSES ARE:
MT03, KA0, MS0, DW0, XEA0, MEA1,
XGA0, TXA, LCA, RHO, DBA4, DBA5,
RH1, DRB0
ENTER DEVICE: MT03
VALID RESPONSES ARE:
TYPE, LINK, GENERIC, CSR, VECTOR, BR
ENTER FIELD: BR
ENTER VALUE: DECIMAL 4 THRU 7:  7
COMMAND: EXIT
CURRENT FILE HAS CHANGES.
SAVE FILE [(Yes), No) YES
ENTER 'WRITE' COMMAND TO SAVE FILE
COMMAND? LI
FILE IN MEMORY 'CONFIG.COM'
!! AUTOMATIC SIZING PROGRAM.
!! THIS IS A TOOL AND NOT A DIAGNOSTIC PROGRAM.
!! CONFIGURATION FILE FOR SYSTEM.
!! COMPUTER GENERATED FOR PROCESSOR SERIAL # 3070 CPU TYPE 1
!! NUMERIC VALUES WITH A LEADING ZERO ARE ASSIGNED.
!! FILE VALID ONLY FOR STANDARD HARDWARE CONFIGURATION.
!! TIME IS   7-JAN-1987 10:46:19.91
!!
```

```
!! DEFINE PROCESSOR...
!!
DS> ATTACH BS16 HUB MT03 767773 773 7
DS> ATTACH KAA40 HUB KA0 YES YES 1FFF 1
!!
!! DEFINE MEMORY...
!!
DS> ATTACH MSA40 HUB MS0 1 1
!!
!!DEFINE SKRBUS ADAPTERS...
!!
DS> ATTACH DWA40 HUB DW0 3 4              SKRBUS I/O SPACE NUMBER  0
!!
DS> ATTACH SNA16 DW0 XEA0 774510 0120 05
DS> ATTACH BM16 DW0 MEA1 772520 0224 05
DS> ATTACH AMF18S DW0 XGA0 760340 0314 05 NONE
DS> ATTACH AMF18A DW0 TXA 760340 0314 05 0377 9600 INTERNAL YES
DS> ATTACH AMF18P DW0 LCA 760340 0314 05 LP
!!
!! DEFINE MSBUS   ADAPTERS...
!!
DS> ATTACH RHA40 HUB RH0 8 5
!!
DS> ATTACH PP06 RHO DBA4
DS> ATTACH PP05 RH0 DBA5
!!
DS> ATTACH RHA40 HUB RH1 9 5
!!
DS> ATTACH PM05 RH1 DRB0
COMMAND? EXI
CURRENT FILE HAS CHANGES.
SAVE FILE? [(Yes), No] NO

.. End of run, 0 errors detected, pass count is 1,
   time is 7-JAN-1987 10:56:36.84
```

12.6.2.3. EXIT

Das Kommando beendet den Test. Die Steuerung wird wieder an den Diagnose-Supervisor übergeben. Wurde nach einem SIZE- oder READ-Kommando eine Information in der generierten Datei gewechselt (CHANGE-Kommando), fragt das Programm, ob die Datei gesichert werden soll.

Format:

EXIT

Nachfolgendes Beispiel zeigt die Wirkung des EXIT-Kommandos ohne und mit vorherigem Dateiwechsel.

```
DS> ST/SE:SE

.. Program: PVSBA _ AUTOSIZER LEVEL 3, revision 1.0, 3 tests,
   at 13:03:38.54

COMMAND? READ DRA0:
CONFIGURATION FILE IS NORMALLY READ/WRITTEN TO THE SUPERVISOR CONSOLE MEDIA
VERIFY SUPERVISOR CONSOLE MEDIA LOADED.
VERIFIED? [(Yes), No] Y
FILE 'DRA0:' READ IN.
COMMAND? EXIT

.. End of run, 0 errors detected, pass count is 1,
   time is 19-JAN-1987 13:04:00.94

DS> ST/SE:SE

.. Program: PVSBA - AUTOSIZER LEVEL 3, revision 1.0, 3 tests,
   at 13:04:06.50.

COMMAND? READ DRA0:
CONFIGURATION FILE IS NORMALLY READ/WRITTEN TO THE SUPERVISOR CONSOLE MEDIA
VERIFY SUPERVISOR CONSOLE MEDIA LOADED.
VERIFIED? [(Yes), No] YES
FILE 'DRA0:' READ IN.
COMMAND? CH
VALID RESPONSES ARE:
KA0, MS0, RH0, DRA0, DRA1, DRA2,
DRA3, DBA4, DBA5, DBA6, DBA7, RH1,
DRB0, DRB1, DRB2, DRB3, DW0, MEA0,
MEA1, MEA2, MEA3, XGA0, TXA, TXA0,
TXA1, TXA2, TXA3, TXA4, TXA5, TXA6,
TXA7, LCA, LCA0
ENTER DEVICE: MEA0:
INVALID OR UNKNOWN DEVICE. 'MEA0:'
ENTER DEVICE: MEA0
VALID RESPONSES ARE:
TYPE, LINK, GENERIC, CSR, VECTOR, BR
ENTER FIELD: GENERIC
ENTER ALPHANUMERIC VALUE.
ENTER VALUE: MEA4
COMMAND? EXIT
CURRENT FILE HAS CHANGES.
SAVE FILE? [(Yes), No] YES
ENTER 'WRITE' COMMAND TO SAVE FILE
COMMAND? WRITE
```

12.6.2.4. HELP

Ohne das Testprogramm zu verlassen, können die verschiedenen Informationen der HELP-Datei PVSBA.HLP angezeigt werden (siehe auch HELP-Kommando des Diagnose-Supervisor).

Format:

```
          HELP [s]
          s ... Schlüsselwort, z.B. SECTION

DS> ST/SE:SE

.. Program: PVSBA - AUTOSIZER LEVEL 3, revision 1.0, 3 tests,
   at 09:35:32.42.

COMMAND? H
AMBIGUOUS COMMAND.
COMMAND? HELP


  HELP

       PVSBA     1.0                              Stand: 22.01.88

       PVSBA ist kein Diagnoseprogramm, sondern ein Hilfmittel.
       PVSBA laeuft innerhalb der Ebene 3 (Stand-alone-Modus). Es er-
       mittelt die aktuelle Konfiguration eines RVS K 1840 im
       Stand-alone-Modus unter Steuerung des Diagnose-Supervisor.
       Das Programm generiert ATTACH-Kommandos fuer alle Geraete der
       jeweiligen Konfiguration des RVS K 1840.
       Bei entsprechender Testauswahl koennen die generierten ATTACH-
       Folgen manuell bearbeitet werden. Diese Datei kann dann als
       Konfigurationskommandodatei zur Abarbeitung der Diagnose-
       programme (auch im User-Modus) genutzt werden.


  Additional information available:

  HELP            DEVICES         EVENT           QUICK           SECTIONS
  SCRIPT          SUMMARY

COMMAND? HELP SEC


  SECTIONS

      Das Programm hat 3 Sektionen


  Additional information available:

  DEFAULT         MANUAL          SELFTEST

COMMAND? EXIT

.. End of run, 0 errors detected, pass count is 1,
   time is  20-JAN-1987 09:36:58.66
DS>
```

12.6.2.5. LIST

Das Kommando ist nur verwendbar nach Ausführung eines SIZE- oder READ-Kommandos. Der Ausgabetext aller Informationen zu den Geräten steht im laufenden Datenpuffer und wird ausgelistet. Die Basis der GENERIC-Namen eines Gerätes wird ausgelistet, wenn die Gerätebezeichnung im Datenpuffer als Teil des Gerätenamens spezifiziert ist.

Beispiel: siehe CHANGE-Kommando

Format:

```
LIST [<device>]
```

12.6.2.6. SIZE

Durch Abarbeitung spezieller Testfolgen wird die aktuelle Gerätekonfiguration bestimmt. Daraus werden mittels umfangreicher Konfigurationstabellen die entsprechenden ATTACH-Folgen generiert, ausgelistet und in einem Datenpuffer als Datei zusammengestellt. Die Abarbeitung durch den Supervisor erfolgt erst mittels des ATTACH-Kommandos dieser Testsektion.
Beispiel: siehe CHANGE-Kommando

Format:

```
SIZE
```

12.6.2.7. READ

Das Kommando liest die spezifizierte Datei vom Ladegerät und speichert die Informationen in den Datenpuffer. Falls keine Datei spezifiziert ist, wird als Dateityp standardmäßig .COM angenommen. Befand sich eine Datei bereits im Datenpuffer, wird diese überschrieben.
Anschließend kann die Datei z.B. durch das CHANGE-Kommando bearbeitet werden.

Format:

```
READ [<file-spec>]
```

Nachfolgendes Beispiel zeigt das Einlesen der Datei CONFIG.COM von der Wechselplatte.

```
DS> ST/SE:SE

   Program: PVSBA - AUTOSIZER LEVEL 3, revision 1.0, 3 tests,
   at 13:00:15.80.

Test 2: SELFTEST MODE
COMMAND? LI
NO FILE IN MEMORY.
COMMAND? REA
CONFIGURATION FILE IS NORMALLY READ/WRITTEN TO THE SUPERVISOR CONSOLE MEDIA
VERIFY SUPERVISOR CONSOLE MEDIA LOADED.
VERIFIED? [(Yes), No] NO
```

```
LOAD SUPERVISOR AND TRY AGAIN.
COMMAND? REA
CONFIGURATION FILE IS NORMALLY READ/WRITTEN TO THE SUPERVISOR CONSOLE MEDIA
VERIFY SUPERVISOR CONSOLE MEDIA LOADED.
VERIFIED? [(Yes), No] YES
FILE '' NOT FOUND.
COMMAND? LI
NO FILE IN MEMORY.
COMMAND? REA
CONFIGURATION FILE IS NORMALLY READ/WRITTEN TO THE SUPERVISOR CONSOLE MEDIA
VERIFY SUPERVISOR CONSOLE MEDIA LOADED.
VERIFIED? [(Yes), No] YES
FILE '' NOT FOUND.
COMMAND? REA 'CONFIG.COM'
CONFIGURATION FILE IS NORMALLY READ/WRITTEN TO THE SUPERVISOR CONSOLE MEDIA
VERIFY SUPERVISOR CONSOLE MEDIA LOADED.
VERIFIED? [(Yes), No] YES
FILE 'CONFIG.COM' NOT FOUND.
COMMAND? REA DRA0:CONFIG.COM
CONFIGURATION FILE IS NORMALLY READ/WRITTEN TO THE SUPERVISOR CONSOLE MEDIA
VERIFY SUPERVISOR CONSOLE MEDIA LOADED.
VERIFIED? [(Yes), No] YES
FILE 'DRA0:CONFIG.COM' READ IN.
COMMAND? LI
FILE IN MEMORY 'CONFIG.COM'
DS> ! Konfigurationsdatei fuer RVS K 1840
DS> ! CONFIG.COM
DS> !
DS> ! CPU, Hauptspeicher
DS> !
DS> ATTACH KAA40 HUB KA0 YES YES 1FFF 1
DS> ATTACH MSA40 HUB MS0 1 0
DS> !
DS> ! MSBUS-Adapter
DS> !
DS> ATTACH RHA40 HUB RH0 8 5
DS> ATTACH PM80 RH0 DRA0
DS> ATTACH PM80 RH0 DRA1
DS> ATTACH PM80 RH0 DRA2
DS> ATTACH PM80 RH0 DRA3
DS> !
DS> ! Wechselplattenspeicher
DS> !
DS> ATTACH PP06 RH0 DBA4
DS> ATTACH PP06 RH0 DBA5
DS> ATTACH PP06 RH0 DBA6
DS> ATTACH PP06 RH0 DBA7
DS> !
DS> ATTACH RHA40 HUB RH1 9 5
DS> !
DS> ! Festplattenspeicher
DS> !
DS> ATTACH PM05 RH1 DRB0
DS> ATTACH PM05 RH1 DRB1
DS> ATTACH PM05 RH1 DRB2
DS> ATTACH PM05 RH1 DRB3
DS> !
DS> ! SKRBUS-Adapter
```

```
DS> !
DS> ATTACH DWA40 HUB DW0 3 4
DS> !
DS> ! Magnetbandgeraete
DS> !
DS> ATTACH BM16 DW0 MEA0 772520 224 5
DS> ATTACH BM16 DW0 MEA1 772520 224 5
DS> ATTACH BM16 DW0 MEA2 772520 224 5
DS> ATTACH BM16 DW0 MEA3 772520 224 5
DS> !
DS> ! Multifunktions-Kommunikationskontroller
DS> !
DS> ATTACH AMF18S DW0 XGA0 760340 314 5 NONE
DS> ATTACH AMF18A DW0 TXA 760340 314 5 377 INTERNAL YES
DS> ATTACH TG911 TXA TXA0
DS> ATTACH TG911 TXA TXA1
DS> ATTACH TG911 TXA TXA2
DS> ATTACH TG911 TXA TXA3
DS> ATTACH TG911 TXA TXA4
DS> ATTACH TG911 TXA TXA5
DS> ATTACH TG911 TXA TXA6
DS> ATTACH TG911 TXA TXA7
DS> !
DS> ! Paralleldrucker
DS> !
DS> ATTACH AMF18P DW0 LCA 760340 314 5 LP
DS> !
DS> ! Terminals
DS> ATTACH DP27 LCA LCA0
COMMAND?
```

12.6.2.8. WRITE

Das Kommando ist nur gültig nach Ausführung eines SIZE- oder READ-Kommandos. Schreiben der aktuellen Datei im Datenpuffer auf die Konsoldiskette. Wenn keine Dateispezifikation übergeben wurde, ist der Standard CONFIG.COM. Ist der Dateiname nicht spezifiziert, lautet der Standard CONFIG und für einen nicht spezifizierten Typ ist der Standard .COM.

Format:

```
WRITE [<file-spec>]
```

Nachfolgendes Beispiel zeigt die Anwendung des WRITE-Kommandos, einschließlich des Versuchs, auf die Wechselplatte zu schreiben.

```
DS> ST/SE:SE

.. Program: PVSBA - AUTOSIZER LEVEL 3, revision 1.0, 3 tests,
   at 13:06:53.96.

COMMAND? REA DRA0:
CONFIGURATION FILE IS NORMALLY READ/WRITTEN TO THE SUPERVISOR CONSOLE MEDIA
VERIFY SUPERVISOR CONSOLE MEDIA LOADED.
VERIFIED? [(Yes), No] YES
```

```
FILE 'DRA0:' READ IN.
COMMAND? WRITE DRA0:
INCORRECT FORMAT.
DRA0:
COMMAND? WRITE
CONFIGURATION FILE IS NORMALLY READ/WRITTEN TO THE SUPERVISOR CONSOLE MEDIA
VERIFY SUPERVISOR CONSOLE MEDIA LOADED.
VERIFIED? [(Yes), No] YES
FILE 'CONFIG.COM' BEING REPLACED.
COMMAND?
```

12.6.3. Test 03: MANUAL MODE
Manual-Modus

Der Test entspricht prinzipiell dem Test 02, nur daß beim SIZE-Kommando die sofortige Auslistung der ATTACH-Folge entfällt.
Nachfolgendes Beispiel erläutert die Anwendung von SIZE, CHANGE, LIST, ATTACH und EXIT-Kommando.

```
DS> ST/SE:MA
.. Program: PVSBA - AUTOSIZER LEVEL 3, revision 1.0, 3 tests,
   at 10:57:14.88.

Test 3: MANUAL MODE
COMMAND? SI
COMMAND? CH
VALID RESPONSES ARE:
KA0, MS0, DW0, XEA0, MEA0, MEA1,
XGA0, TXA, LCA, RH0, DBA4, DBA5,
RH1, DRB0
ENTER DEVICE: MEA0
VALID RESPONSES ARE:
TYPE, LINK, GENERIC, CSR, VECTOR, BR
ENTER FIELD: LINK
VALID RESPONSES ARE:
HUB, KA0, MS0, DW0, XEA0, MEA1,
XGA0, TXA, LCA, RH0, DBA4, DBA5,
RH1, DRB0
ENTER DEVICE: HUB
COMMAND? LI
FILE IN MEMORY 'CONFIG.COM'
!! AUTOMATIC SIZING PROGRAM.
!! THIS IS A TOOL AND NOT A DIAGNOSTIC PROGRAM.
!! CONFIGURATION FILE FOR SYSTEM.
!! COMPUTER GENERATED FOR PROCESSOR SERIAL # 3070 CPU TYPE 1
!! NUMERIC VALUES WITH A LEADING ZERO ARE ASSIGNED.
!! FILE VALID ONLY FOR STANDARD HARDWARE CONFIGURATION.
!! TIME IS   7-JAN-1987 10:57:26.07
!!
!! DEFINE PROCESSOR...
!!
DS> ATTACH BM16 HUB MEA0 772520 0224 05
DS> ATTACH KAA40 HUB KA0 YES YES 1FFF 1
!!
!! DEFINE MEMORY...
!!
```

```
DS> ATTACH MSA40 HUB MS0 1  1
!!
!! DEFINE SKRBUS ADAPTERS...
!!
DS> ATTACH DWA40 HUB DW0 3 4            !! SKRBUS I/O SPACE NUMBER  0
!!
DS> ATTACH SNA16 DW0 XEA0 774510 0120 05
DS> ATTACH BM16 DW0 MEA1 772520 0224 05
DS> ATTACH AMF18S DW0 XGA0 760340 0314 05 NONE
DS> ATTACH AMF18A DW0 TXA 760340 0314 05 0377 9600 INTERNAL YES
DS> ATTACH AMF18P DW0 LCA 760340 0314 05 LP
!!
!! DEFINE MSBUS   ADAPTERS...
!!
DS> ATTACH RHA40 HUB RH0 8 5
!!
DS> ATTACH PP06 RH0 DBA4
DS> ATTACH PP05 RH0 DBA5
!!
DS> ATTACH RHA40 HUB RH1 9 5
!!
DS> ATTACH PM05 RH1 DRB0
COMMAND? ATT
COMMAND? EXI

.. End of run, 0 errors detected, pass count is 1,
   time is 7-JAN-1987 10:58:46.02
DS> SH DEV
_CSA0    console  HUB    00000000
_RH1     RHA40    HUB    60012000  TR=9. BR=5.
_DRA0    PM05     _RH1   60012400
_KA0     KAA40    HUB    00000000  G-floating instructions=Yes H-floating instruc
tions=Yes WCS last address=00001FFF(X) Accelerator type=1.
_MS0     MSA40    HUB    60002000  TR=1. Number of arrays=1.
_DW0     DWA40    HUB    60006000  TR=3. BR=4.
_XEA0    SNA16    _DW0   6013F948  CSR=774510(O) VECTOR=000120(O) BR=5.
_MEA0    BM16     _DW0   6013F550  CSR=772520(O) VECTOR=000224(O) BR=5.
_XGA0    AMF18S   _DW0   6013E0E0  CSR=760340(O) Vector=000314(O) BR=5. External
Wrap=NONE
_TXA     AMF18A   _DW0   6013E0E0  CSR=760340(O) Vector=000314(O) BR=5. Active Li
nes=000377(O) Baud Rate=9600 Loopback Type=INTERNAL SKRBUS Init Jumper=Yes
_LCA     AMF18P   _DW0   6013E0E0  CSR=760340(O) Vector=000314(O) BR=5. Port Devi
ce=LP
_RH0     RHA40    HUB    60010000  TR=8. BR=5.
_DBA4    PP06     _RH0   60010600
_DBA5    PP05     _RH0   60010680
_DRB0    PM05     _RH1   60012400
_MEA0    BM16     HUB    0003F550  CSR=772520(O) VECTOR=000224(O) BR=5.
DS>
```

12.7. Auflistung der Tests

Test 01: DEFAULT MODE
Standard-Modus
Test 02: SELFTEST MODE
Eigentest-Modus
Test 03: MANUAL MODE
Manual-Modus

13. PVCAA MSBUS-Adaptertest

13.1. Zusammenfassung

Das Diagnoseprogramm PVCAA (Dok.Nr.: 1.57.700009.1) ist ein Programm der Ebene 3 (standalone Modus). Es ist bestimmt für den Test des MSBUS-Adapters K 2816 (RHA40) des RVS-Rechnersystemes Robotron K 1840.
Das Programm ist nur in Verbindung mit dem Diagnose-Supervisor arbeitsfähig.
Eine Überprüfung des MSBUS-Adapters im online Modus ist mit dem Test 19 möglich.
Hierbei muss ein MSBUS-Exerciser (MBE) zur Verfügung stehen.

13.2. Testbedingungen

13.2.1. Hardware

Die minimalste Hardwarekonfiguration besteht aus:

- Konsolsubsystem	KSS	K 2811
- Zentrale Verarbeitungseinheit	CPU	K 2810
- Hauptspeicher 2 Mbyte	MEM	K 3581.10
- MSBUS-Adapter	MBA	K 2816

Zusätzlich kann der MSBUS-Exerciser (MBE) eingesetzt werden.

13.2.2. Software

Das Programm PVCAA ist nur in Verbindung mit dem Diagnose-Supervisor (DS) im standalone Modus lauffähig.

13.3. Voraussetzungen

Für eine Abarbeitung des Programmes wird die volle Funktionsfähigkeit aller Elemente der CPU vorausgesetzt. Programme der Ebenen 4 und 5 müssen zuvor erfolgreich gelaufen sein.

13.4. Bedienungshinweise

Die Bedienungsanleitung des DS ist für den Lade- u. Startvorgang mitzuverwenden.
Der DS wird vom entsprechenden Gerät geladen und meldet sich mit:

DS>

Das Testprogramm PVCAA.EXE ist zu laden:

```
DS> LOAD FVCAA
```

Vor dem Programmstart muß der Adapter eingegliedert (ATTACH) und für den Test logisch ausgewählt werden (SELECT).

Für RHA40:

```
DS> ATTACH
Device type ?  RHA40
Device link ?  HUB o. ISB
Device name ?  RHn  (z.B. RH0)
TR number ? (dezimal 1-15) Standard <8>
BR level  ? (dezimal 4-7)  Standard <5>
```

Die Übertragungsanforderungsstufe (TR) und die Busanforderungsstufe (BR) sind zusätzliche Informationen für eine korrekte Geräteeingliederung.

Log. Auswahl: DS> SELECT RHn z.B. RH0

Programmstart: DS> START

Weitere Bedienungshinweise sind der DS-Beschreibung zu entnehmen, z.B. die Auswahl bestimmter Testsektionen (s. Abschn. 13.5.13.) über den DS.

13.5. Funktionsbeschreibung

13.5.1. Übersicht

Das Programm beinhaltet 20 Tests. Eine Auflistung aller Tests ist im Abschnitt 13.8. zu finden.
Das Programm wird in drei Objekt-Module unterteilt:

Modul 1:	RHA40_HEADER	
Modul 2:	RHA40_TESTS	(Test 1 bis Test 14)
Modul 3:	RHA40_TESTS	(Test 15 bis Test 20)

13.5.2. Programmgröße

Der Speicherbedarf für dieses Programm beträgt 40056 byte (88 Pages).

Speicheraufteilung des Programmes: %X200 - %XB200

Aufteilung einzelner Sektionen

Sektion	Speicherplätze	Funktion der Sektion
Modul 1:		
BUFFERS	%X2000 - %X25FF	dreimal 512 byte für Dateneingabepuffer, Datenausgabepuffer,MIO-Puffer
CLEANUP	%X2600 - %X267C	Abarbeitung bei jeder Programmbeendigung
DISPATCH	%X2680 - %X285F	Zusammenstellung von Daten, die
DISPATCH_X	%X2A00 - %X2F51	während der Programmbearbeitung aufgerufen werden
HEADER_CODE	%X2A00 - %X2F51	Parameter, Unterprogramme
INITIALIZE	%X2F54 - %X3189	Abarbeitung bei Programmstart
SUMMARY	%X318C - %X3195	Zusammenfassung der Testergebnisse seit Programmstart
Modul 2:	%X3200 - %X6F5C	
Modul 3:	%X7000 - %XB0A8	

13.5.3. Programmlaufzeit

Die Programmlaufzeit für einen fehlerfreien Standarddurchlauf (Sektion DEFAULT) beträgt ca. 42 s.

13.5.4. Abarbeitungsbedingungen

Entfällt.

13.5.5. Flagverwendung

Entfällt.

13.5.6. Allgemeine Programmanforderungen

Entfällt.

13.5.7. Fehlerermittlung

Durch die MBA-Diagnose wird eine Fehlermitteilung generiert. Sie ist abhängig vom Typ des gefundenen Fehlers und des fehlerbehafteten Testes. Jedoch in allen Fällen wird der fehlerbehaftete Modul und die Art des Fehlers identifiziert. Wenn notwendig werden Soll- und Istdaten ausgeschrieben. Es wird zwischen Hard-Fehlern, Gerätefehlern u. Systemfehlern unterschieden. Auf die Hard-Fehler wird nachfolgend näher eingegangen.
Der Umfang der Fehlerausschriften kann mit den DS-Flags IE1, IE2, IE3 und IES beeinflusst werden (s. DS-Beschreibung).

Fehlernachrichtenkopf:

```
(1) | ********  PVCAA RHA40 DIAGNOSTIC - 1.0  ********
(2) | Pass 1, test 16, subtest 15, error 1,   15-DEC-1986
(3) | Hard error while testing RH0: FAILING-MODULE: MIR(8502)
```

Erläuterung:

(1) Überschrift mit Programmname und Revisionsnummer
(2) Durchlaufnummer, Test-, Subtest- u. Fehlernummer
(3) Fehlerart, fehlerbehaftete Module (Steckeinheiten)

13.5.7.1. Register-Hard-Fehler

Erfaßt sind Fehler der internen MBA-Register, der MAP-Register, externer Register u. der Register des MSBUS-Exercisers (MBE).
Beispiel:

```
(1) | ( RHSR) = 40003182 ERROR:
(2) |  EXPECTED:  NRCONF, DT_COMP, DTABT, MBEXC, IS_TIMEOUT, RD_TIMEOUT
(3) |  RECEIVED:  NRCONF, DT_COMP, DTABT, MXF, MBEXC, IS_TIMEOUT
(4) |       XOR:  MXF, RD_TIMEOUT
(5) |  EXPECTED:  40003083  RECEIVED:  40003182  XOR:  00000101
```

Erläuterung:

(1) Bezeichnung des Registers in dem ein Fehler auftrat (RHxx = xx Register des MBA)
(2) Sollwert
(3) Istwert
(4) Differenz (XOR Vergleich)
(5) Darstellung der Bitmnemoniken

Die in den Auschriften erscheinenden Bitabkürzungen werden im Abschnitt 13.6.21. näher beschrieben.

13.5.7.2. Map-Hard-Fehler

Erfaßt werden Fehler der Funktion des Mappings.
Beispiel:

```
(1) | MAP FUNCTION ERROR
(2) | EXPECTED MAP = 80000050      MAP SELECTED = 80000040
(3) | PHYSICAL ADDRESS EXPECTED: 00002800  ADDRESS CONSTRUCTED: 00002700
```

Erläuterung:

(1) Fehlername
(2) Inhalt des Map-Registers welches ausgewählt werden sollte u. Inhalt des aktuell ausgewählten Map-Registers
(3) physische Sollwert- und physische Istwertadresse

13.5.7.3. Datenvergleichs-Hard-Fehler

Erfaßt werden Datenvergleichsfehler.

Beispiel:

```
(1) | DATA COMPARE ERROR:
(2) | EXPECTED:: 00  RECEIVED:: 10
(3) | EXP PHYS ADDRESS:: 0002204  REC PHYS ADDDRESS:: 00002204
      BYTE NUMBER:: 4
```

Erläuterung:

(1) Fehlername
(2) Soll- und Istwert
(3) physische Sollwert- und physische Istwertadresse

13.5.8. Programmitteilungen

Alle Programmitteilungen erfolgen über DS-Routinen.
Folgende Ausschriften erscheinen bei jedem Programmdurchlauf, wenn die entsprechenden Bedingungen erfüllt sind.

```
Test  9:  NO DRIVE RESPONDED
          --> kein Laufwerk angeschlossen

          DRIVE [n] RESPONDED
          --> Laufwerk n ist angeschlossen.

Test 19:  ******** NO MBE PRESENT ON MSBUS ********
          --> kein MBE vorhanden
```

Basisadressen: S. Abschn. 13.5.11..

Zusätzlich werden Programmitteilungen nach einigen Fehlermitteilungen generiert.

```
Beispiele: - RETURN PC = 00005E85
           - ISB ERROR: CSR = <fehlerhaftes Bitmnemonik>
```

13.5.9. Leistungsparamter des Testobjektes

Entfällt.

13.5.10. Aktionen bei unerwarteten Fehlern

Interrupts und Exceptions, die nicht vom Diagnoseprogramm initiiert werden, entstehen als Folge von unerwarteten Fehlern. Wurde eine Exception über den Vektor %X04 (machine check) erkannt, erfolgt die dementsprechende selbsterklärende Mitteilung.
Die Programmfortführung ist u.U. mit DS>CONTINUE <CR> möglich.

13.5.11. Testfolge bei mehreren Einheiten

Wenn durch den Operator mehr als ein MBA ausgewählt wurde, wird die Basisadresse für den jeweiligen MBA ausgegeben und der Test wird auf diesem MBA ausgeführt.
Nachdem der erste Durchlauf für alle Untertests abgeschlossen wurde, wird eine Endnachricht für den ersten Durchlauf ausgegeben. Sind durch den Operator weitere Durchläufe definiert worden, wird die Mitteilung über die Basisadresse für alle folgenden Durchläufe unterdrückt.

13.5.12. Einstellungen für Ablauf unter AFT

Entfällt.

13.5.13. Testsektionen

Das Programm PVCAA enthält drei Testsektionen.

DEFAULT : Abarbeitung aller Tests.

ALL : Abarbeitung aller Tests.

MBE : Nur der Test 19 wird abgearbeitet.
Wird im Test 19 festgestellt, daß kein MSBUS-Exerciser am MSBUS vorhanden ist, so wird vom Programm
******** NO MBE PRESENT ON MSBUS ********
ausgegeben.
Zusätzlich wird bei gesetztem QA-Flag die Testausschrift "RH11-TB tests EXECUTION NOT POSSIBLE" statt "RH11-TB tests" dargestellt.

13.6. Testbeschreibung

Die in der Testbeschreibung erwähnten Register werden im Abschnitt 13.6.21. erläutert.

13.6.1. Test 1 - Interner Busbestätigungstest

Dieser Test bestätigt, daß der interne Bus und der 8-1 Tristate Multiplexer die Übertragung der Logikpegel 0 und 1 in allen Bitpositionen auf dem internen Register-Modul (MIR) des MBA gewährleisten.
Durch die Verwendung von Bitmasken (%XF u. %XFFFFFF28) wird versucht, Bitpositionen im CR bzw. CSR zu löschen. Treten Haftfehler des Logikpegels 1 auf, erfolgen Fehlernachrichten für Fehler 1 u. Fehler 2 des Subtestes 0. Es erfolgt eine Neuinitialisierung des MBA und des MSBUS. Im Bytezähler-Register (BCR), Diagnose-Register (DR) und Map-Register 0 werden alle Bitpositionen mit 1 beschrieben. Diese Register werden nacheinander in das Register R3 übertragen. Mittels Komplementbildung wird überprüft, ob Haftfehler des Logikpegels 0 in den Registern auftreten. Ist dies der Fall, erfolgt die Fehlernachricht für Subtest 0, Fehler 3.
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.2. Test 2 - Initialisierungstest

Dieser Test überprüft, ob in den internen Registern des MBA Haftfehler des Logikpegels 1 auftreten. Es erfolgt die Initialisierung des MBA. Alle internen Register werden gelöscht, rückgelesen und auf Haftfehler des Logikpegels 1 überprüft.

Überblick über zu prüfende Register, Prüfvorgang, Fehlernummer und mögliche fehlerbehaftete BLP:

Register	Prüfvorgang	Fehler	BLP
CSR	Das CSR wird gelesen und auf einen korrekten Adapter-Kode (Bit 5 gesetzt) überprüft.	1	MIR(8502)
CSR	Im CSR werden alle Bitpositionen mit 0 beschrieben. Das Register wird gelesen und auf Haftfehler des Logikpegels 1 geprüft.	2	MIR(8502)
SR	Das SR wird gelöscht, gelesen und auf Haftfehler des Logikpegels 1 geprüft.	3	MSI(8501) MIR(8502)
CR	Das CR wird gelöscht, gelesen und auf Haftfehler des Logikpegels 1 geprüft.	4	MIR(8502)
DR	Lesen des DR. Das MBDIB-Feld wird gelöscht und auf Haftfehler des Logikpegels 1 geprüft.	5	MIR(8502) MDP(8503)
VAR	Alle Bitpositionen werden mit 1 beschrieben, das Register wird gelöscht und auf Haftfehler des Logikpegels 1 geprüft.	6	MIR(8502)
BCR		7	MIR(8502)
Map-Reg. (256)	Das jeweilige Map-Reg. wird gelöscht und auf Haftfehler des Logikpegels 1 geprüft.	8	MIR(8502)

13.6.3. Test 3 - Steuerregistertest

Dieser Test überprüft das Steuerregister (CR), welches ein Lese-/Schreibregister ist, auf Haftfehler der Logikpegel 0 und 1.
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.3.1. Subtest 1

Alle Bitpositionen des CR werden mit 1 beschrieben. Das Register wird gelöscht und auf Haftfehler des Logikpegels 1 überprüft.

13.6.3.2. Subtest 2

In diesem Test erfolgt die Überprüfung für drei Bitzuordnungen des CR. Die Register R3 und R4 dienen als Istwertregister (CR-Inhalt) bzw. als Sollwertregister.
Im CR werden die Prüfbits gesetzt, das CR wird in R3 gelesen und mit R4 verglichen. Bei Differenzen erfolgt die Fehlernachricht für den jeweiligen Fehler des Subtest 2.

Fehlernummer	Prüfbit
1	Bit 3 = Maintenance-Modus
2	Bit 2 = Interrupt-Freigabe
3	Bit 1 = Abbruch der Übertragung

13.6.4. Test 4 - Diagnose-Registertest

Dieser Test überprüft das Diagnose-Register (DR) auf Haftfehler der Logikpegel 1 u. 0. Dieses Lese-/Schreibregister gestattet es, eine Diagnose des MBA ohne angeschlossene Laufwerke durchzuführen.

13.6.4.1. Subtest 1

Das DR wird auf allen Bitpositionen mit 1 beschrieben. Das Byte 0 (MBDIB-Feld) des DR wird gelöscht und gelesen. Sind hier noch Bits gesetzt, erfolgt die Fehlernachricht.
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.4.2. Subtest 2

Im CR wird Bit 3 (Maintenance-Modus) gesetzt. Im DR wird Byte 0 gelöscht und das DR wird gelesen. Wurde Bit 20 (MSBUS FAIL) nicht gesetzt, erfolgt die Fehlernachricht.
Mögliche fehlerbehaftete BLP: MIR(8502) und MCP(8504).

13.6.4.3. Subtest 3

Im DR werden alle beschreibaren Bits mit 1 beschrieben. Das DR wird gelöscht und gelesen. Wurde das READ/WRITE-Feld des DR (Bit 21 bis Bit 31) nicht gelöscht, wird die Fehlernachricht ausgegeben.
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.4.4. Subtest 4

In das DR werden nacheinander 10 Masken eingetragen, wobei jeweils eine Maske einem Bitwert entspricht (27, 31, 30, 29, 28, 27, 20, 25, 24, 23). Wird bei dem Einschreiben nicht das entsprechende Bit gesetzt, erfolgt die Fehlernachricht.
Mögliche fehlerbehaftete BLP: MIR(8502).
Das DR wird gelöscht. Nachfolgend wird das Ansprechen folgender Bits des DR überprüft.

Fehlernummer	Bitwert	Bitfunktion	fehlerbehaftete BLP
2	20	MSBUS FAIL	MCP(8504)
3	16	MSBUS CTOD	MCP(8504)
4	19	MSBUS RUN	MDP(8503)
5	17	MSBUS EXC	MCP(8504)

Anschließend werden MSBUS-Laufwerkauswahlfeld (Bit <15:13>) und MSBUS-Registerauswahlfeld (Bit <12:08>) überprüft.
Mögliche fehlerbehaftete BLP: MDP(8503).

13.6.4.5. Subtest 5

Ein Datenübertragungsabbruch wird simuliert, wobei das DR auf Löschen des Bit 19 (MSBUS RUN) getestet wird .
Mögliche fehlerbehaftete BLP: MIR(8502) und MCP(8504).

13.6.5. Test 5 - Test des Registers für virtuelle Adresse

Dieser Test überprüft das Register für die virtuelle Adresse (VAR) des MBA auf Haftfehler des Logikpegels 0. Die Register R3 und R4 dienen als Ist- bzw. als Sollwertregister. Für das Testen wird eine 1 durch Linksverschiebung von Bit 0 bis Bit 16 transportiert. Bei jeder Bitposition erfolgt eine Linksverschiebung von Bit 0 bis Bit 16. Sind Bits nicht gesetzt worden, erfolgt die Fehlernachricht.
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.6. Test 6 - Map-Registertest

Dieser Test überprüft die 256 Map-Register des MBA auf Haftfehler des Logikpegels 0 und duale Adressierungsfehler. Die Register R3 und R4 dienen als Istwert- bzw. Sollwertregister.
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.6.1. Subtest 1

Alle 256 Map-Register werden überprüft, inwieweit sich die Maske %X801FFFFF in den Registern setzen läßt. Die Maske wird in R4 und in das jeweilige Map-Register geschrieben. Beide Register werden über die Exklusiv-Oder-Funktion miteinander verglichen.

13.6.6.2. Subtest 2

Dieser Subtest überprüft die Bits <20:00> (physikalische Page-Frame-Anzahl) der 256 Map-Register auf Haftfehler des Logikpegels 0.
Für die Testung wird eine 1 durch Linksverschiebung auf alle Bitpositionen gesetzt. Der jeweilige gesetzte Wert wird mit dem Sollwert verglichen. Bei einer Differenz erfolgt die Fehlerausschrift für Fehler 1.
Anschließend wird in jedem Map-Register das gültige Eintragen des Gültigkeitsbits (Bit 31) überprüft (Fehler 2).

13.6.6.3. Subtest 3

Dieser Subtest überprüft alle 256 unbeschriebenen Map-Register auf duale Adressierungsfehler.

13.6.7. Test 7 - Test der Datenwege externer Register

Dieser Test prüft den MSBUS-Steuerweg auf Haftfehler der Logikpegel 0 und 1. Desweiteren erfolgt ein Testen auf Steuerwegparitätsfehler. Die Register R3 und R4 dienen als Istwert- bzw. als Sollwertregister.
Das Laufwerkregister wird gelöscht. Durch Rücklesen wird auf Haftfehler des Logikpegels 1 geprüft (Fehler 1).
Anschließend wird nacheinander jede Bitposition (16 Bit) durch Linksverschiebung mit einer 1 beschrieben. Durch Rücklesen des Registers wird auf Haftfehler des Logikpegels 0 geprüft (Fehler 2).
Weiterhin erfolgt ein Testen auf einen MSBUS-Steuerwegparitätsfehler. Bit 17 ist im SR gesetzt (Fehler 3).
Mögliche fehlerbehaftete BLP: MCP(8504).

13.6.8. Test 8 - MBDIB-Diagnose-Registertest

Dieser Test überprüft die Integrität der MBA-Datenwege. In den einzelnen Subtests werden die Dateneingabe- u. die Datenausgabepuffer (MBDIB u. MBDOB) zu je 512 byte unterschiedlich beschrieben, wobei im Datenausgabepuffer die zu erreichenden Sollwerte stehen. Eine Ausnahme bildet der Subtest 2, wo nur 8 byte beschrieben werden.
Durch Setzen entsprechender Bits im DR wird eine Datenübertragung vollzogen. Anschließend erfolgt ein Vergleich der zwei Datenpuffer. Nacheinander werden die 512 byte des MBDOB und die 512 byte des MBDIB in R4 (Sollwert) bzw. in R3 eingetragen. Bestehen Differenzen zwischen den Puffern, erfolgt eine Fehlernachricht.
Mit den DS-Flags Quick und Operator kann die Anzahl der Fehlerausschriften beeinflußt werden.

Quick-Flag	Operat.-Flag	Wirkung
0	0	Maximal acht gefundene Fehler werden aufgelistet.
1	0	Am Ende jedes Subtestes erfolgt die Angabe der
1	1	Gesamtfehlersumme.
0	1	Auflistung aller auftretenden Fehler.

Anfangsbelegungen für den MBDOB und den MBDIB für die einzelnen Subtests:

Subtest	MBDIB	MBDOB
1	1 auf allen Bitpositionen (512 byte).	0 auf allen Bitpositionen (512 byte).
2	0 auf allen Bitpositionen (512 byte).	1 auf allen Bitpositionen (512 byte).
3	0 auf allen Bitpositionen (512 byte).	1 auf allen Bitpositionen (512 byte).
4	-"-	Auf jedes Langwort der 512 byte wird eine Bitmaske geschrieben, die durch Linksverschiebung einer 1 von Bit 0 bis Bit 31 erzeugt wird.
5	-"-	S. Subtest 4, statt einer 1 wird eine 0 verwendet.
6	-"-	Die Maske %XF00F wird auf 256 Wörter eingeschrieben.
7	-"-	Die Maske %XA55A wird auf 256 Wörter eingeschrieben.

Die Register sind belegt mit:

R3 = Istwert
R4 = Sollwert
R6 = physische Sollpufferadresse
R7 = physische Istpufferadresse
R9 = Bytezahl

Mögliche fehlerbehaftete BLP: MDP(8503).

13.6.9. Test 9 - Statusregistertest

Dieser Test überprüft die folgenden Bits des SR auf Haftfehler des Logikpegels 0:

Bit 31 = Datenübertragung belegt
Bit 19 = Programmierfehler
Bit 18 = Laufwerk existiert nicht
Bit 16 = Attention-Signal vom MSBUS
Bit 13 = vollständige Datenübertragung
Bit 12 = Abbruch der Datenübertragung
Bit 11 = verspätete Datenübertragung
Bit 8 = vermißte Datenübertragung
Bit 7 = MSBUS-Exception

13.6.9.1. Subtest 1

Im CR und DR werden Bits gesetzt, die bei einer Datenübertragung zu Veränderungen im SR führen. Das SR wird in R3 (Istwert) gelesen und R3 wird auf gesetztes Sollbit getestet. Ist das jeweilige Bit nicht gesetzt worden, erfolgt die entsprechende Fehlernachricht (Fehler 1 bis 4). Im R4 stehen die entsprechenden Sollbits.
Anschließend wird abgefragt, ob nach Löschen des SR noch Bits gesetzt sind (Fehler 5).
Bit- und Registerzuordnungen für die Fehler 1 bis 4:

Fehlernr.	Bits im CR	Bits im DR	Bits im SR	Bits im R4
1	/	25 (Simul. Occupied)	31	31
2	1	26 (Simul. EBL)	12	7, 12, 13
3	1	-"-	7	-"-
4	1	-"-	13	-"-

Mögliche fehlerbehaftete BLP: MIR(8502), MDP(8503) und MCP(8504).

13.6.9.2. Subtest 2

Dieser Subtest überprüft das Setzen des Bit 16.
Durch Setzen des Bits 24 (Simulation Attention) im DR wird versucht, das Bit 16 im SR zu setzen (Fehler 1). Sind nach dem Löschen des SR noch Bits gesetzt, erfolgt die Fehlernachricht für Fehler 2.
Mögliche fehlerbehaftete BLP: MIR(8502) und MCP(8504).

13.6.9.3. Subtest 3

Dieser Subtest wird nicht im Maintenance-Modus abgearbeitet. Es wird nacheinander auf acht Laufwerke zugegriffen und geprüft, welches Laufwerk vorhanden ist. Bei jedem Zugriff wird das SR gelesen und auf Bit 18 geprüft (Fehler 1). Fehler 2 s. Subtest 2. Sind keine Laufwerke angeschlossen, wird am Ende des Subtestes die Mitteilung

NO DRIVE RESPONDED

ausgegeben.
Ansonsten erfolgt die Ausschrift für das entsprechende angeschlossene Laufwerk [n]

DRIVE [n] RESPONDED

Mögliche fehlerbehaftete BLP: MIR(8502) und MCP(8504).

13.6.9.4. Subtest 4 und Subtest 5

Diese Subtests überprüfen das Setzen von Bit 8 (Subtest 4) und Bit 11 (Subtest 5) für den Fehlerfall 1. Im R4 sind die Sollbits 8, 12, 13 bzw. 7, 11, 12, 13 gesetzt.

Subtest 4: READ-Kommando -> simulieren von OCC und SCLK im DR -> vermißte Datenübertragung

Subtest 5: WRITE-Kommando -> leeres SILO -> simulieren von SCLK im DR -> verspätete Datenübertragung

Sind nach Löschen des SR weitere Bits gesetzt, erfolgt die Fehlernachricht für Fehler 2.
Mögliche fehlerbehaftete BLP: MIR(8502), MCP(8504) bzw. MSI(8501), MIR(8502).

13.6.9.5. Subtest 6 bis 9

In den folgenden vier Subtests wird das Setzen des Bit 19 überprüft. Dies wird in den einzelnen Subtests für verschiedene Arbeitsweisen vollzogen.

Subtestnr.	Arbeitsweise
6	Versuch einer Übertragung bei bereits vorhandener, dabei Aussenden eines Übertragungskomm.
7	Versuch einer Übertragung bei bereits vorhandener, dabei Versuch des Löschens des BCR.
8	Versuch einer Übertragung bei beereits vorhandener, dabei Versuch des Löschens des VAR.
9	Versuch einer Übertragung bei bereits vorhandener, dabei Versuch des Löschens eines Map-Reg..

Im R4 sind die Sollbits 31 und 19 gesetzt. Das SR wird gelesen und auf gesetztes Bit 19 geprüft. Fehler 2 s. Subtest 2.
Mögliche fehlerbehaftete BLP: MIR(8502) und MCP(8504).

13.6.9.6. Subtest 10

Es wird eine Datenübertragung ausgeführt, wobei im DR Bit 28 (Blocksendekommando zum ISB) gesetzt wurde. Damit wird eine verspätete Datenübertragung und ein Abbruch der Datenübertragung erzwungen. Das SR wird auf gesetztes Bit 11 getestet.
Mögliche fehlerbehaftete BLP: MSI(8501), MIR(8502) und MDP(8503).

13.6.9.7. Subtest 11

Es wird versucht während einer Datenübetragung in den Maintenance-Modus überzugehen. Das SR wird gelesen und auf gestztes Bit 19 getestet.
Mögliche fehlerbehaftete BLP: MIR(8502) und MCP(8504).

13.6.10. Test 10 - Test der Mapping-Funktion

Dieser Test überprüft die korrekte Auswahl der Map-Register, die durch das VAR angezeigt werden.

13.6.10.1. Subtest 1

Dieser Subtest überprüft, ob die Auswahl des Map-Registers, welches durch das VAR angezeigt wurde, fehlerfrei erfolgt.
In den Registern steht folgender Inhalt:

R3 = Inhalt des tatsächlich ausgewählten Map-Registers
R4 = Inhalt des Map-Registers, welches ausgewählt werden sollte (Sollwertregister)
R6 = Map-Registerinhalt ohne Bit 31
R7 = ISB Kommandoteil des CAR, wodurch Map-Register falsch ausgewählt wurde
R8 = physische Sollwertadresse
R9 = tatsächlich ausgewählte physische Adresse

Jeder Map-Registerwert wird gleich zu seinem Einerkomplement (Basis 256) gesetzt, z.B. Map-Register 0 = Map-Register 255. Über ein Unterprogramm (ISSUE_MMREAD) wird die Übergabe in den Maintenance-Modus ermöglicht. Ein READ-Kommando wird zu einem Pseudolaufwerk gesendet, wobei ein bestätigter Programmdurchlauf gewährleistet wird.
Das R7 wird mit der Maske %XB verglichen. Besteht keine Übereinstimmung (falsche ISB-Kommandoadresse), erfolgt die Fehlernachricht für Fehler 1.
Durch einen Vergleich der physischen Adressen wird die richtige Map-Registerauswahl überprüft (Fehler 2). Diese zwei Prüffolgen erfolgen für alle 256 Map-Register.
Mögliche fehlerbehaftete BLP: MSI(8501) und MIR(8502).

13.6.10.2. Subtest 2

Der Subtest 2 überprüft, inwieweit Map-Paritätsfehler auftreten.
Dabei sind R3 und R4 belegt mit:

R3 = Istwertregister
R4 = Sollwertregister, gesetztes Bit 31 (belegte Datenübertragung)

Es werden 21 Map-Register (0 bis 20) mit einer 1 in fortlaufender Bitposition (Bit 0 -> Bit 20) und Bit 31 beschrieben. Das Map-Register 21 wird auf allen Bitpositionen mit einer 1 beschrieben. Im Map-Register 22 wird Bit 31 gesetzt. Für jedes Map-Register wird folgende Prüffolge durchgeführt. Das BCR wird auf einen willkürlichen Wert gesetzt, im DR werden die Bits 25 u. 28 gesetzt. Das SR wird gelesen. Besteht zum R4 eine Abweichung, erfolgt die Fehlernachricht.
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.11. Test 11 – Übertragungstest für das Statusregister

Dieser Test überprüft, daß für die folgenden Bits des SR keine Haftfehler des Logikpegels 0 auftreten.

Bit 30 = keine Antwortbestätigung
Bit 17 = MSBUS-Steuerparitätsfehler
Bit 10 = höherer Schreibkontrollfehler
Bit 9 = niederer Schreibkontrollfehler
Bit 6 = MSBUS-Datenparitätsfehler
Bit 5 = Map-Paritätsfehler
Bit 4 = ungültiges Map
Bit 3 = Fehlerbestätigung
Bit 1 = Interface-Folge-Time-Out
Bit 0 = READ-Daten-Time-Out

Die Register R3 u. R4 dienen als Istwertregister (Inhalt des SR) bzw. als Sollwertregister (außer Subtest 1).

13.6.11.1. Subtest 1

Bildung einer Pseudo-NEXUS-Adresse. Die Time-Out-Service-Routine wird aufgerufen und nachfolgend das Time-Out-Ereignisflag gesetzt. Erfolgt kein Time-Out (zurückgeführter PC Wert gleich Null), dann wird eine Fehlernachricht ausgegeben.
Mögliche fehlerbehaftete BLP: MSI(8501).

13.6.11.2. Subtest 2 bis Subtest 9

In den Subtests 2 bis 9 werden Programmschritte ausgeführt, die zu Bitbelegungen im SR führen. Das SR wird in R3 gelesen und auf das jeweilige Sollwertbit (in R4 ebenfalls gesetzt) geprüft. Wenn das Bit nicht gesetzt wurde, erfolgt die Fehlernachricht für Fehler 1 des jeweiligen Subtest.
Das SR wird gelöscht und auf noch gesetzte Bits geprüft (Fehler 2, außer bei Subtest 9, dann Fehler 4).

13.6.11.2.1. Subtest 2

Durch Lesen von Daten vom MSBUS zum Speicher und des Setzens des Bit 31 im DR (invertierte MSBUS-Steuerparität) wird ein MSBUS-Datenparitätsfehler erzwungen.
Gesetzte Bits im R4: 6, 7, 12, 13
Gesetztes Bit im SR: 6
Mögliche fehlerbehaftete BLP: MIR(8502), MDP(8503)

13.6.11.2.2. Subtest 3

Im DR wird Bit 30 gesetzt (invertierte MSBUS-Steuerparität). Durch Lesen des Laufwerkregisters 0 wird ein MSBUS-Steuerparitätsfehler erzwungen.
Gesetzte Bits im R4: 7, 12, 13, 17
Gesetztes Bit im SR: 17
Mögliche fehlerbehaftete BLP: MIR(8502), MCP(8504)

13.6.11.2.3. Subtest 4

Ein Bestätigungsfehler wird durch eine Datenübertragung vom Kontrollerspeicher zum MSBUS erzwungen.
Gesetzte Bits im R4: 3, 7, 12, 13
Gesetztes Bit im SR: 3
Mögliche fehlerbehaftete BLP: MSI(8501), MIR(8502)

13.6.11.2.4. Subtest 5

Ein Map-Paritätsfehler wird durch Setzen des Bit 29 (invertierte Map-Parität) erzwungen.
Gesetzte Bits im R4: 5, 7, 12, 13
Gesetztes Bit im SR: 5
Mögliche fehlerbehaftete BLP: MIR(8502)

13.6.11.2.5. Subtest 6

Ein Map-Paritätsfehler wird durch eine Datenübertragung zum MSBUS, unter Verwendung eines ungültigen Map erzwungen.
Gesetzte Bits im R4: 4, 7, 12, 13
Gesetztes Bit im SR: 4
Mögliche fehlerbehaftete BLP: MIR(8502)

13.6.11.2.6. Subtest 7

Ein niederer Schreibkontrollfehler wird durch das Schreiben von Daten während des Maintenance-Modus erwungen.
Gesetzte Bits im R4: 7, 9, 12, 13
Gesetztes Bit im SR: 9
Mögliche fehlerbehaftete BLP: MIR(8502), MDP(8503)

13.6.11.2.7. Subtest 8

Ein höherer Schreibkontrollfehler wird durch eine WRITE-Prüfung mit falschen Datenmasken erzwungen.
Gesetzte Bits im R4: 7, 10, 12, 13
Gesetztes Bit im SR: 10
Mögliche fehlerbehaftete BLP: MIR(8502), MDP(8503)

13.6.11.2.8. Subtest 9

Durch den Versuch einer Datenübertragung von einem nichtexistierenden Speicher, wird ein READ-Daten-Time-Out-Fehler erzwungen.
Gesetzte Bits im R4: 0, 1, 7, 12, 13, 30
Gesetzte Bits im SR: 1 (Fehler 1), 30 (Fehler 2), 0 (Fehler 3)
Mögliche fehlerbehaftete BLP: MSI(8501), MIR(8502)

13.6.12. Test 12 - Blocküberträgungstest

Dieser Test überprüft die korrekte Datenübertragung vom MSBUS zum Speicher und die Datenübertragung über die Page-Grenzen hinaus.

13.6.12.1. Subtest 1

In das erste und letzte Langwort eines 1-Page-Puffers wird eine Datenmaske (1 auf allen Bitpositionen) eingelesen. Die virtuelle Adresse wird dann so gesetzt, daß sie auf die letzte Langwortgrenze in der Page zeigt. Die ausgeführte Übertragung verursacht, daß das VAR das nächste Map-Register auswählt und vier Byte in das erste Langwort des Puffers übeträgt. Das SR wird gelesen und auf korrekten Status (Bit 13 gesetzt, d.h. Datenübertragung vollständig) geprüft.
Mögliche fehlerbehaftete BLP's: MIR(8502) und MDP(8503).

13.6.12.2. Subtest 2

Dieser Subtest überprüft die korrekte Datenübertragung über die Page-Grenzen hinaus. Das Testen erfolgt für vollzogene Datenübertragung (Fehler 1), korrektes erstes Langwort (Fehler 2) und korrektes letztes Langwort des Puffers (Fehler 3).
Mit dem Quick-Flag des DS kann der Umfang der Fehlerausschriften verringert werden. Bei gesetztem Quick-Flag werden maximal 10 Fehler aufgelistet. Bei nicht gesetztem Quick-Flag werden alle Fehler aufgelistet (maximal 512).
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.13. Test 13 - Bytezählertest

Dieser Test überprüft das BCR.

13.6.13.1. Subtest 1

Dieser Subtest prüft das BCR auf Haftfehler des Logikpegels 0. Auf 16 Bitpositionen des BCR (Bit 0 bis Bit 15) wird nacheinander eine 1 geschrieben. Das BCR wird jedesmal gelesen und auf einen Haftfehler geprüft. Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.13.2. Subtest 2

Dieser Subtest dient für die Festlegung der SILO-Größe (16 o. 256 byte), und setzt für die nächsten zwei Subtests den jeweiligen Größenwert fest, wenn sie auf die SILO-Ggröße zurückgreifen.
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.13.3. Subtest 3

Die niedere Hälfte des BCR (ISB-Bytezähler) wird getestet (WRITE-Kommando). Bei jedem empfangenen Byte muß diese Hälfte des BCR decrementiert werden. Treten hierbei Fehler auf, erfolgt die Fehlernachricht.
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.13.4. Subtest 4

Die höhere Hälfte des BCR (MSBUS-Bytezähler) wird getestet (READ-Kommando). Diese Hälfte des BCR muß bei jedem Byte, welches vom MSBUS im SILO empfangen wird, decrementiert werden. Erfolgt dies nicht, wird die Fehlernachricht ausgegeben.
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.14. Test 14 - READ/WRITE-Funktionstest

Dieser Test überprüft die Page-Übertragung des MBA mit einer korrekten Ausführung von READ, READ FORWARD, WRITE CHECK und WRITE CHECK REVERSE.

13.6.14.1. Subtest 1

Alle 256 Map-Register werden beschrieben. Ein WRITE-Kommando wird zum Laufwerk 0 gegeben. Durch Setzen und Rücksetzen des DR-Bit 25 (Simulation von Occupied), Bit 27 (Simulation von SCLK) und Bit 26 (Simulation von EBL) erfolgt eine Datenübertragung . Das SR wird gelesen und auf gesetztes Bit 13 (Datenübertragung vollständig) geprüft (Fehler 1). Im R4 wurde ebenfalls Bit 13 gesetzt.
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.14.2. Subtest 2 bis Subtest 5

Es erfolgt die Page-Übertragung mit jedem der folgenden Kommandos, unter Nutzung aller 256 Map-Register.

Subtestnummer	Kommando
2	READ FORWARD
3	READ REVERSE
4	WRITE CHECK REVERSE
5	WRITE CHECK

Es wird durch Lesen des SR auf vollständige Datenübertragung geprüft. Ist Bit 13 (Datenübertragung vollständig) nicht gesetzt, erfolgt die Fehlernachricht für Fehler 1 des jeweiligen Subtestes.
Anschließend wird das VAR gelesen und auf einen korrekten Map-Pointer geprüft (Fehler 2).
Der Datenpuffer wird gelesen. Zuvor wurde eine Komplementbildung von Null durchgeführt. Sind nicht alle Bitpositionen mit 1 beschrieben, erfolgt die Fehlernachricht für Fehler 3.
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.15. Test 15 - SILO-Test

Dieser Test überprüft, daß das SILO leer ist und keine Daten von ihm zum Speicher übertragen werden, wenn ein Fehler während der Übertragung auftritt.

13.6.15.1. Subtest 1

Dieser Subtest überprüft die Ausführung einer Datenübertragung. Nach der Simulation der Datenübertragung wird das SR gelesen und auf gesetztes Bit 13 (Datenübertragung vollständig) geprüft.
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.15.2. Subtest 2

Zu dem Laufwerk 0 wird ein READ-Kommando gegeben. Sechs synchrone Takte werden auf den MSBUS gegeben (Bit 27 im DR Setzen, Rücksetzen). Im DR wird weiterhin das Bit 31 (invertierte MSBUS-Datenparität) gesetzt, wodurch ein Abbruch der Datenübertragung erzwungen wird (Fehler 1).
Anschließend wird das DR gelesen und auf gesetztes Bit 17 (MSBUS-Exception) geprüft (Fehler 2).
Mögliche fehlerbehaftete BLP: MDP(8503) und MCP(8504).

13.6.16. Test 16 - Interrupttest

Dieser Test überprüft, daß für jede Statusbedingung, die eine Interruptgenerierung bewirkt, die Interrupts auch auftreten. Die Register R3 und R4 dienen als Istwert- bzw. als Sollwertregister.
Bitzuordnungen des SR:

Bit 31 = Datenübertragung belegt
Bit 30 = keine Antwortbestätigung
Bit 19 = Programmierfehler
Bit 18 = Laufwerk existiert nicht
Bit 13 = vollständige Datenübertragung
Bit 12 = Abbruch der Datenübertragung
Bit 11 = verspätete Datenübertragung
Bit 10 = höherer Schreibkontrollfehler
Bit 9 = niederer Schreibkontrollfehler
Bit 8 = vermißte Datenübertragung
Bit 7 = MSBUS-Exception
Bit 6 = MSBUS-Datenparitätsfehler
Bit 4 = ungültiges Map
Bit 3 = Fehlerbestätigung
Bit 1 = Interface-Folge-Time-Out
Bit 0 = READ-Daten-Time-Out

Die Interrupt-Freigabe wird durch Setzen des Bit 2 im CR ermöglicht. Damit wird dem MBA ein Interrupt an die CPU erlaubt, wenn die entsprechenden Bedingungen gegeben sind. Weiterhin wird im internen Prozessorregister (IPLR) die Interruptebene %X14 eingetragen.

13.6.16.1. Subtest 1

Mit diesem Subtest wird der Datenübertragungsinterrupt überprüft.
Auf Laufwerk 0 wird ein READ-Kommando gegeben und eine Datenübertragung simuliert. Das SR wird in R3 gelesen und auf gesetztes Bit 31 (DT BUSY) geprüft (Fehler 1).
Durch Setzen des Bits 1 im CR (Datenübertragungsabbruch) wird ein Interrupt provoziert. In R4 werden die Bits 7, 12 und 13 gesetzt. Das SR wird wieder gelesen und auf gesetztes Interruptflag (Bit 0) geprüft (Fehler 2). Das Flag wird

gelöscht.
Es erfolgt ein Vergleich von R3 und R4 auf korrekten Status (Fehler 3).
Die wiederhergestellte Datenübertragung wird durch Lesen des SR auf gesetztes Bit 31 geprüft (Fehler 4).
Mögliche fehlerbehaftete BLP: MIR(8502) und MCP(8504).

13.6.16.2. Subtest 2

Dieser Subtest überprüft den Attention-Interrupt.
Eine Interruptprovokation erfolgt, so daß durch Setzen des Bits 24 im DR (Simulation ATT) das Attention-Signal vom MSBUS simuliert wird. In R4 ist Bit 16 gesetzt. Das SR wird gelesen und auf gesetztes Interruptflag geprüft (Fehler 1).
Das Flag wird gelöscht und das SR auf einen korrekten Status geprüft (Fehler 2).
Das SR wird gelöscht und auf noch gesetzte Bits geprüft (Fehler 3).
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.16.3. Subtest 3

Dieser Subtest überprüft den Interrupt für nicht vorhandene Laufwerke (Bit 18).

Achtung: Dieser Interrupt kann nicht getestet werden, wenn acht Laufwerke angeschlossen sind.

Das Testen erfolgt nacheinander für acht Laufwerke. Zu dem jeweiligen Laufwerk wird ein NOOP (0 auf allen Bitpositionen) gesendet. Das SR wird gelesen und auf Bit 18 geprüft. Ist es gelöscht, so wird der Interrupttest nicht ausgeführt (Laufwerk ist wirklich nicht angeschlossen). Ansonsten wird ein Interrupt provoziert und das SR auf gesetztes Interruptflag getestet.
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.16.4. Subtest 4

Mit diesem Subtest wird der Interrupt für vermißte Datenübertragung geprüft (Bit 8).
In R4 sind die Bits 8, 12, 13 gesetzt. Durch Nichtsetzen von Occupied (Bit 25 im DR) wird ein Interrupt provoziert. Das SR wird auf gesetztes Interruptflag getestet (Fehler 1).
Das Flag wird gelöscht. Die Register R3 und R4 werden miteinander auf einen korrekten Status geprüft (Fehler 2).
Die wiederhergestellte Datenübertragung wird durch Lesen des SR auf gesetztes Bit 31 geprüft (Fehler 3).
Mögliche fehlerbehaftete BLP: MIR(8502) und MCP(8504).

13.6.16.5. Subtest 5

Dieser Subtest überprüft den MSBUS-Exception-Interrupt (Bit 8). In R4 sind die Bits 7, 11, 12, 13 gesetzt.
Durch Setzen des Bit 28 (Blocksendekommando zum ISB) und 257-maliges Setzen/Rücksetzen von SCLK im DR wird eine verspätete Datenübertragung und ein Abbruch der Datenübertragung erzwungen. Dadurch wird ein MSBUS-Exception-Signal erzeugt. Nach vollzogener Interruptprovokation wird das SR auf gesetztes Interruptflag getestet (Fehler 1).
Es erfolgt eine Testung auf korrekten Status (Fehler 2).
Die wiederhergestellte Datenübertragung wird durch Lesen des SR auf gesetztes

Bit 31 geprüft (Fehler 3).
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.16.6. Subtest 6

Mit diesem Subtest wird der Programmierfehler-Interrupt überprüft.
Der Interrupt wird durch Auslösen einer Datenübertragung (READ) bei bereits vorhandener Datenübertragung (WRITE) verursacht. Das SR wird gelesen und auf gesetztes Interruptflag geprüft (Fehler 1).
Das Flag wird gelöscht. Es erfolgt das Testen auf korrekten Status (Fehler 2).
Nachdem das SR gelöscht wurde, wird es nochmals gelesen und auf noch gesetzte Bits getestet (Fehler 3).
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.16.7. Subtest 7

Dieser Subtest prüft den Interrupt für Datenverspätung, wenn vom Laufwerk gelesen wurde, das SILO voll ist und synchrone Takte auf dem MSBUS empfangen wurden.
Im DR wird das Bit 28 (Blocksendekommando zum ISB) gesetzt, damit wird eine Datenverspätung erzwungen. Das SR wird gelesen und auf gesetztes Interruptflag getestet (Fehler 1).
Das Flag wird gelöscht. Es erfolgt das Testen auf korrekten Status (Fehler 2) und wiederhergestellte Datenübertragung (Fehler 3).
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.16.8. Subtests 8, 9 und 11

Diese Subtests überprüfen die Interrupts für Paritätsfehler. Es wird jeweils ein Interrupt durch Setzen des jeweiligen Bits für invertierte Parität erzwungen.
In R4 werden die Sollbits gesetzt. Das SR wird gelesen und auf gesetztes Interruptflag getestet (Fehler 1).
Es erfolgt dasTesten auf korrekten Status (Fehler 2). Nachdem das SR gelöscht wurde, wird es gelesen und auf noch gesetzte Bits geprüft (Fehler 3).

Subtest	Paritätsfehler	Bits im R4	Bits im DR
8	MSBUS-Datenparitätsfehler	6, 7, 12, 13	31
9	MSBUS-Steuerparitätsfehler	7, 12, 13, 17	30
11	MAP-Paritätsfehler	5, 7, 12, 13	29

Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.16.9. Subtest 10

Dieser Subtest überprüft den Interrupt für die Fehlerbestätigung bei einem Time-Out.
Im R4 sind die Bits 3, 7, 12, 13 gesetzt. Ein Interrupt wird durch Erhalt einer Fehlerbestätigung auf ein WRITE Kommando erzwungen. Das SR wird gelesen und auf gesetztes Interruptflag getestet (Fehler 1).
Es wird der korrekte Status (Fehler 2) und die wiederhergestellte Datenübertragung überprüft (Fehler 3).
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.16.10. Subtest 12

Dieser Subtest überprüft den Interrupt bei ungültigem Map.
Ein Interrupt wird provoziert durch eine Datenübertragung mit ungültigem Map (Gültigkeitsbit ist gelöscht). Das SR wird auf gesetztes Interruptflag geprüft (Fehler 1).
Weiterhin wird der korrekte Status überprüft (Fehler 2).
Im R4 sind dafür die Bits 4, 7, 12 und 13 gesetzt. Die wiederhergestellte Datenübertragung wird durch Lesen des SR auf gesetztes Bit 31 geprüft (Fehler 3).
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.16.11. Subtest 13 und Subtest 14

Diese Subtests überprüfen die Schreibkontrollfehler.
In R4 sind die Bits 7, 9, 12, 13 (Subtest 13) und 7, 10, 12, 13 gesetzt. Ein Fehler wird im höheren Byte bzw. niederen Byte erzwungen. Das SR wird gelesen und auf gesetztes Interruptflag geprüft (Fehler 1).
Der korrekte Status wird geprüft (Fehler 2), ebenfalls die wiederhergestellte Datenübertragung durch Lesen des SR auf gesetztes Bit 31 (Fehler 3).
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.16.12. Subtest 15

Dieser Subtest überprüft den Interrupt für ein READ-Daten-Time-Out.
Im R4 sind die Bits 0, 1, 7, 12, 13, 30 gesetzt. Ein Interrupt wird durch ein READ-Daten-Time-Out provoziert. Das SR wird auf gesetztes Interruptflag getestet (Fehler 1), korrekten Status (Fehler 2) und wiederhergestellte Datenübertragung (Fehler 3).
Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.17. Test 17 - Erweiterter MCP-Test

Dieser Test überprüft zusätzlich die MCP-Logik, welche in den anderen Tests nicht überprüft wird.

13.6.17.1. Subtest 1

Dieser Subtest überprüft das Nichtsetzen des Bit 8 (vermißte Datenübertragung) im SR, wenn die Datenübertragung vollzogen wird und 600 synchrone Takte auf dem MSBUS gegeben wurden. Wird es gesetzt, erfolgt die Fehlernachricht.
Mögliche fehlerbehaftete BLP: MCP(8504).

13.6.17.2. Subtest 2

Dieser Subtest überprüft das Attention-Signal vom MSBUS, Bit 16 muß im SR gesetzt sein. Erfolgt dies nicht, wird die Fehlernachricht ausgegeben.
Mögliche fehlerbehaftete BLP: MCP(8504).

13.6.18. Test 18

<reserviert>

13.6.19. Test 19 - RH11-TB Tests

Dieser Test überprüft das Vorhandensein eines MSBUS-Exercisers (MBE). Wenn kein MBE vorhanden ist, wird der Test abgebrochen, ansonsten erfolgt eine Prüfung auf korrekte Initialisierung des MBE, Integrität aller MSBUS-Leitungen und Datenübertragungen mit sehr hoher Geschwindigkeit.
Die Register R3 und R4 dienen als Istwert- u. Sollwertregister.

13.6.19.1. Subtest 1

Es erfolgt eine Kontrolle auf Vorhandensein eines MBE. Ist dies nicht der Fall, wird die Mitteilung

******** NO MBE ON MSBUS ********

ausgegeben und der Test 19 wird beendet. Zusätzlich wird bei gesetztem QA-Flag die Fehlerausschrift
"RH11-TB tests EXECUTION NOT POSSIBLE !" statt "RH11-TB tests" dargestellt.
Im weiteren Ablauf werden Bitzuordnungen im MBE_CR1 und MBE_SR überprüft. Die Sollbits sind in R4 gesetzt und die zu kontrollierenden Register werden in R3 übertragen.

Fehlernr.	Register	Prüfbits	
1	MBE_CR1	Bit 11 - Laufwerkverfügbarkeit	
2	MBE_SR	Bit 15 - Attention aktiv	) dürfen
		Bit 14 - Fehler	) nicht
		Bit 13 - Positionierung im Lauf	) gesetzt sein
3	MBE_SR	Bit 12 - online	) müssen
		Bit 8 - Laufwerk angeschlossen	) gesetzt
		Bit 7 - Laufwerk lesend	) sein

Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.19.2. Subtest 2

Mit diesem Subtest werden die MSBUS-Leitungen (DS <02:00>) für die Laufwerkauswahl überprüft. Dies erfolgt für die Laufwerke 0 bis 7.
Ein Laufwerk wird immer ausgewählt. Das SR wird gelesen und auf Bit 18 überprüft (Laufwerk nicht vorhanden). Ist das Laufwerk nicht vorhanden, wird der MBE für dieses Laufwerk ausgewählt, Bit 18 wird gelöscht und das MBE_ASR wird gelesen. Ist dessen Laufwerkauswahlfeld unkorrekt, erfolgt die Fehlernachricht.
Maske für das Laufwerkauswahlfeld des MBE_ASR = %XFFFF8FFF.
Mögliche fehlerbehaftete BLPs: MIR(8502) und MDP(8503).

13.6.19.3. Subtest 3

Mit diesem Subtest werden die MSBUS-Leitungen für die Registerauswahl überprüft RS <04:00>).
Die Laufwerkregister werden nacheinander ausgewählt. Jedesmal wird das MBE_ASR gelesen und auf ein korrektes Registerauswahlfeld überprüft. Bei einer Differenz erfolgt die Fehlernachricht.
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.19.4. Subtest 4

Mit diesem Subtest wird die MSBUS-Attention-Leitung überprüft.
In das MBE_CR1 wird ein SEARCH-Kommando geschrieben. Damit wird im MBE der Attention-Status geschaffen. Das MBE_SR wird gelesen und auf gesetztes Bit 15 (Attention aktiv) geprüft. Ist dieses Bit nicht gesetzt, erfolgt die Fehlernachricht.
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.19.5. Subtest 5

Mit diesem Subtest werden die MSBUS-Steuerleitungen (C <15:00>) überprüft.
In R4 und MBE_DBR werden Datenmasken eingeschrieben. Das MBE_DBR wird gelesen und auf auftretende Haftfehler der Logikpegel 0 u. 1 überprüft.

Fehlernummer	Datenmaske
1	%X0000FFFF
2	%X00000000
3	1 nacheinander von Bit 0 bis Bit 15
4	0 -"-

Das MBE_ER wird gelesen und auf Bit 3 (Steuerparitätsfehler) geprüft. Ist es gesetzt, erfolgt die Fehlernachricht (Fehler 5).
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.19.6. Subtest 6

Dieser Subtest überprüft die MSBUS-Occupied-Leitung.
Im MBE_MR wird die Negation von Occupied (Bit 8) und Diagnose-Modus (Bit 10) gesetzt. Es erfolgt eine Datenübertragung. Das SR wird gelesen und auf Bit 8 geprüft. Ist dieses Bit nicht gesetzt, erfolgt die Fehlernachricht für Fehler 1.
Mögliche fehlerbehaftete BLP: MIR(8502). und MDP(8503).

13.6.19.7. Subtest 7

Mit diesem Subtest wird die MSBUS-FAIL-Leitung überprüft.
Dieser MSBUS-Ausfall wird durch die Übergabe des MBA in den Maintenance-Modus erzwungen. Das MBE_ER wird gelesen und auf Bit 6 (MSBUS FAIL) überprüft. Ist dieses Bit nicht gesetzt, erfolgt die Fehlernachricht für Fehler 1.
Nach einer Neuinitialisierung wird das MBE_ER nochmals gelesen. Sind noch weitere Bits gesetzt, wird die Fehlernachricht für Fehler 2 ausgegeben.
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.19.8. Subtest 8 bis Subtest 15

Mit diesen Subtests werden Datenübertragungen mit verschiedenen Bitmasken überprüft.
Der MBDIB und der MBDOB werden zu je 512 byte beschrieben. Ein WRITE-Kommando wird gegeben und das MBE_SR wird auf Bit 14 (Fehler) geprüft. Ist Bit 14 gesetzt, erfolgt die Fehlernachricht für Fehler 1.
Sind weitere Bits gesetzt, erfolgt die Fehlernachricht für Fehler 2.
Ein READ-Kommando wird gegeben. Das SR wird gelesen und auf korrekten Status (Fehlermaske = %X319E0) getestet (Fehler 3).
Nach der Datenübertragung wird das MBDIB mit dem MBDOB verglichen. Bei einer Differenz erfolgt die Fehlernachricht für Fehler 4.
Insgesamt erfolgt diese Nachricht maximal acht mal. Am Ende eines jeden Subtestes wird die tatsächliche Fehleranzahl des Fehlers 4 angegeben.

Subtestnr.	MBDOB
8	1 auf allen Bitpositionen (128 Langwörter)
9	0 auf allen Bitpositionen -"-
10	Bitmaske %XF00F (256 Wörter)
11	Bitmaske %XA55A -"-
12	Bitmaske %XAA55 -"-
13	1 in fortlaufender Bitposition durch Linksverschiebung von Bit 0 bis Bit 31 (256 Wörter)
14	0 in fortlaufender Bitposition durch Linksverschiebung von Bit 0 bis Bit 31 (256 Wörter)
15	1 auf allen Bitpositionen, umgekehrte WRITE-Prüfung, ungerade Bytezahl

Das MBDIB wurde am Anfang bei allen Subtests auf allen Bitpositionen der 128 Langwörter mit 0 beschrieben.
Mögliche fehlerbehaftete BLP: MIR(8502) und MDP(8503).

13.6.20. Test 20 - Test des Ladens des VAR

Dieser Test überprüft das korrekte Laden des VAR. Die Register R3 und R4 dienen als Istwertregister (Inhalt des VAR) und als Sollwertregister (Datenmaske). R4 und VAR werden mit Datenmasken beschrieben (24 insgesamt). Das VAR wird in R3 gelesen. Bei einer Differenz zwischen R3 und R4 erfolgt die Fehlernachricht für Fehler 1.

Datenmasken: %X778, %X80, %X6F8, %X100, %X5F8, %X200, %X3F8, %X400, %X7780, %X800, %X6F80, %X1000, %X5F80, %X2000, %X3F80, %X4000, %X17800, %X8000, %XF800, %X10000, %X17FF8, %X8000, %XFFF8, %X100000

Mögliche fehlerbehaftete BLP: MIR(8502).

13.6.21. Registerbeschreibung

13.6.21.1. Interne Prozessorregister

IPLR = Interrupt-Prioritätsebenen-Register
SBIER = ISB-Fehlerrgister
SBIFS = ISB-Fehlerstatusregister

13.6.21.2. MBA interne Register

CSR - Konfigurations-/Statusregister (Read/Write)

PC-Offset zur MBA-Basisadresse = %X00.
Bitzuordnungen mit Abkürzungen für Fehlerausschriften:

<31> - ISB-Paritätsfehler (ISBPE)
<30> - Schreibdatenfolgefehler (WS)
<29> - nichtprogrammierter Lesedatenfehler (URD)
<28> - 0 reserviert
<27> - Mehrfachtransmitter (MT)
<26> - Transmitterfehler (XMTFLT)
<25:24> - 0 reserviert
<23> - Adapterspannung "AUS" (PD)
<22> - Adapterspannung "EIN" (PU)
<21:08> - 0 reserviert
<07:00> - Adapter-Kode, Bit 5 ist gesetzt

CR - Steuerregister (Read/Write)

PC-Offset zur MBA-Basisadresse: %X04.
Bitzuordnungen mit Abkürzungen für Fehlerausschriften:

<31:04> - reserviert
<03> - Maintenance-Modus (MMM)
<02> - Interrupt-Freigabe (IE)
<01> - Datenübertragungsabbruch (ABORT)
<00> - Initialisierung des MBA und des MSBUS (INIT)

SR - Statusregister (Read/Write)

PC-Offset zur MBA-Basisadresse: %X08.
Bitzuordnungen mit Abkürzungen für Fehlerausschriften:

<31> - Datenübertragung belegt (DT BUSY)
<30> - keine Antwortbestätigung (NRCONF)
<29> - korrigierte Lesedaten (CRD)
<28:20> - 0 reserviert
<19> - Programmierfehler (PGE)
<18> - Laufwerk existiert nicht (NED)
<17> - MSBUS-Steuerwegparitätsfehler (MCPE)
<16> - Attention-Signal vom MSBUS (ATTN)
<15:14> - 0 reserviert
<13> - vollständige Datenübertragung (DT_COMP)
<12> - Abbruch der Datenübertragung (DTABT)
<11> - verspätete Datenübertragung (DLT)
<10> - höherer Schreibkontrollfehler (WCK_UP_ERR)
<09> - niederer Schreibkontrollfehler (WCK_LOW_ERR)
<08> - vermißte Datenübertragung (MXF)
<07> - MSBUS-Exception (MBEXC)
<06> - MSBUS-Datenparitätsfehler (MDPE)
<05> - MAP-Paritätsfehler (MAPPE)
<04> - ungültiges Map (INVMAP)
<03> - Fehlerbestätigung (ERR_CONF)
<02> - Lesedatenersatz (RDS)
<01> - Interface-Folge-Time-Out (IS_TIMEOUT)
<00> - READ-Daten-Time-Out (RD_TIMEOUT)

VAR - Register für virtuelle Adresse (Read/Write)

PC-Offset zur MBA-Basisadresse: %X0C.
Bitzuordnungen:

<31:17) - 0
<16:09> - Map-Pointer (Auswahl eines der 256 Map-Register)
<08:00> - physikalische Page-Byte-Adresse

BCR - Bytezähler-Register (Read/Write)

PC-Offset zur MBA-Basisadresse: %X10.
Bitzuordnungen:

<31:16> - MSBUS-Bytezähler (nur Read)
<15:00> - ISB-Bytezähler

DR - Diagnose-Register (Read/Write)

PC-Offset zur MBA-Basisadresse: %X14.
Bitzuordnungen mit Abkürzungen für Fehlerausschriften:

<31> - Generierung der invertierten Datenparität (IMDPG)
<30> - Generierung der invertierten Steuerparität (IMCPG)
<29> - invertierte Map-Parität (IMAPP)
<28> - Blocksendekommando zum ISB (BLKCOM)
<27> - simuliert SCLK (MSBUS-Synchronisierungstakt) (SIMSCLK)
<26> - simuliert EBL (MSBUS-Blockende-Signal) (SIMEBL)
<25> - simuliert OCC (MSBUS-Occupied-Signal) (SIMOCC)
<24> - simuliert ATT (MSBUS-Attention-Signal) (SIMATTN)
<23> - MSBUS-Dateneingabepuffer (MBDIBSEL)
<22:21> - für Wartungszwecke
<20> - MSBUS FAIL (nur Read) (FAIL)
<19> - MSBUS RUN (nur Read) (MRUN)
<18> - MSBUS WCLK (nur Read), Schreibtakt (MWCLK)
<17> - MSBUS EXC (nur Read), Exception während einer Datenübertragung (MEXC)
<16> - MSBUS CTOD (nur Read), Datenübertragungsrichtung (CTOD)
<15:13> - MSBUS-Laufwerkauswahlfeld (nur Read) (MDSn)
<12:08> - MSBUS-Registerauswahlfeld (nur Read) (MRSn)
<07:00> - ausgewähltes MBDIB-Byte (MBDIBn)

MSR - Ausgewähltes-Map-Register (Read)

PC-Offset zur MBA-Basisadresse: %X18.
Bitzuordnungen:

<31> - Gültigkeitsbit
<30:21> - 0 reserviert
<20:00> - repräsentiert physikalische Page-Frame-Anzahl

CAR - Kommando-/Adress-Register (Read)

PC-Offset zur MBA-Basisadresse: %X1C.
Bitzuordnungen:

<31:00> - Wert der Bits <31:00> des ISB während des Kommandoadressteils der nächsten Datenübertragung

13.6.21.3. MBE-Register

MBE_ASR = MBE-Attentionsammel-Register
MBE_CR1 = MBE-Steuerregister 1
MBE_CR2 = MBE-Steuerregister 2
MBE_DBR = MBE-Datenpuffer-Register
MBE_DTR = MBE-Drivertyp-Register
MBE_ER = MBE-Fehler-Register
MBE_MR = MBE-Maintenance-Register
MBE_SR = MBE-Statusregister

13.7. Abkürzungsverzeichnis

BR	- Busanforderungsstufe
CR	- Carriage Return (Wagenrücklauf)
DS	- Diagnose-Supervisor
MBA	- MSBUS-Adapter
MBDIB	- Dateneingabepuffer
MBDOB	- Datenausgabepuffer
MBE	- MSBUS-Exerciser
MCP	- MBA-Steuerweg (Steckeinheit)
MDP	- MBA-Datenkanal (Steckeinheit)
MIR	- MBA-Register (Steckeinheit)
MSI	- MBA-Synchronbus-Interface (Steckeinheit)
PC	- Befehlszähler
TR	- Übertragungsanforderungsstufe

13.8. Auflistung der Tests

Test 1:	Interner Busbestätigungstest
Test 2:	Initialisierungstest
Test 3:	Steuerregistertest
Test 4:	Diagnose-Registertest
Test 5:	Test des Registers für virtuelle Adresse
Test 6:	Map-Registertest
Test 7:	Test der Datenwege externer Register
Test 8:	MBDIB-Diagnose-Registertest
Test 9:	Statusregistertest
Test 10:	Test der Mapping-Funktion
Test 11:	Übertragungstest für das Statusregister
Test 12:	Blockübertragungstest
Test 13:	Bytezählertest
Test 14:	READ/WRITE- Funktionstest
Test 15:	SILO Test
Test 16:	Interrupttest
Test 17:	Erweiterter MCP-Test
Test 18:	<reserviert>
Test 19:	RH11-TB Tests
Test 20:	Test des Ladens des VAR

14. Test des SKRBUS-Adapters

14.1. Zusammenfassung

Das Diagnoseprogramm PVCBA (Dokumentationsnr. 1.57.700010.6) ist ein Programm der Ebene 3 (es arbeitet im Stand-alone-Modus unter der Steuerung des Diagnose-Supervisors). Es ist bestimmt für den Test des SKRBUS-Adapters K 2815 des RVS-Rechnersystems K 1840. Es testet die SKRBUS-Adapter-Logik auf ihre Funktionen und Fehlerbedingungen mit Ausnahme der wenigen, die sich nicht testen lassen. Es gibt 54 Tests, die bei ordnungsgemäßem Durchlauf aufgelistet werden. Im Fehlerfall erscheinen die Pass- (Anzahl der Programmabarbeitungen), Test-, Subtest- und Fehlernummern sowie die Steckeinheitenbezeichnung. Es können maximal 4 SBA angeschlossen sein. Das Testprogramm kann durch die Verwendung der TR-Level (Übertragungsanforderungsstufe) zwischen ihnen unterscheiden.

14.2. Testbedingungen

14.2.1. Hardware

Voraussetzungen für den Programmtest sind :
- Konsolterminal bzw. Bediengerät BDE K 8911.80
- CPU KAA40
- Systemplatte oder Floppy-Laufwerk
- SKRBUS-Adapter DWA40 robotron K 2815
- wahlweise SET

14.2.2. Software

Der SBA-Test läuft unter der Steuerung des Diagnose-Supervisors PSSAA (im Anschluß nur als Supervisor bezeichnet).

14.3. Voraussetzungen

Vor dem Testbeginn müssen die CPU und die Hardware ausführlich getestet sein d.h. die Mikrodiagnose und der Befehlstest müssen erfolgreich getestet sein. Die Voraussetzungen für die einzelnen Tests sind bei der jeweiligen Testbeschreibung aufgeführt (siehe 14.6.xx).

14.4. Bedienungshinweise

Voraussetzung geladener Supervisor

```
DS>ATT DWA40 HUB DWn 3 4;eingliedern der Hardware, die benötigt wird.
DS>SELECT DWn           ; n=0...3 Anzahl der SKRBUS-Adapter
DS>RUN PVCBA            ;oder:
DS>LOAD PVCBA           ;PVCBA wird von der Platte geladen
DS>START/switches       ;und gestartet
```

Die möglichen Schalter (switches) sind in der Supervisorbeschreibung aufgeführt.

14.5. Funktionsbeschreibung

14.5.1. Übersicht

Das Programm PVCBA beinhaltet 54 Tests (siehe Punkt 14.8.) Diese Tests überprüfen die SBA-Register auf ihre Adressierbarkeit, ihren Anfangszustand sowie auf Lesbarkeit und Schreibbarkeit. Die Map-Register, Data-Path-Register und Buffer-Selection-Verification-Register werden auf eindeutige und korrekte Adressierbarkeit geprüft. Es werden RAM-Adresstests, Energieausfall- und Interrupt-Tests (d.h. Test der SKRBUS-Power-Fail-Logik und der Adapter-Interrupt-Logik) sowie der Test der Fehlerbedingungen, d.h. richtiges Setzen und Löschen der Fehlerbits, durchgeführt. Der SBA unterbricht den ISB nach einem SBA-Fehler.
Außerdem beinhaltet PVCBA den Test aller Datenübertragungsmöglichkeiten (DATI, DATI/P, DATO, DATO/B) von und zum ISB bzw. von und zum SKRBUS, sowie die Unterbrechbarkeit des ISB durch den SET in allen Ebenen. Ein ausführlicher Datenübertragungstest wird für alle MAP-Register, Data- Path-Register, die gepufferten Datenwege und die direkten Datenwege, die SKRBUS-Adress- und Datenleitungen durchgeführt und, besonders wichtig, der größte Teil der Mikro-Controller wird getestet.
Das Programm ist in die Module PVCBA1 bis PVCBA5 eingeteilt.

14.5.2. Programmgröße

Der SBA-Test belegt im Haupt-Speicher den Bereich von 200%X bis D63F%X, (53 Kbyte) wobei die Tests 1 bis 54 im Bereich von 2C42%X bis CF00%X liegen. In dem Bereich vor dem ersten Test liegen :
HEADER, TSTCNT, BLANK, ARGLIST, CLEANUP, DATA, DATA.CMP, DISPATCH, DISPATCH_X,INITIALIZE, MCHKSR, PRINT DATA, RXBUF, SUMMARY.
Nach dem letzten Test liegen:TMPBUF,TXBUF.
Der HEADER enthält die Parameter, die zur Programmsteuerumg für den Supervisor dienen. Er wird zu Beginn jedes Durchlaufs ausgeführt.
Die DISPATCH-Tabelle ist eine Zusammenstellung von Adressen, die zu Beginn jedes Tests generiert werden. Das wird durch die Definition der Programmsektion 'DISPATCH' ausgeführt.
Die Programmsektion DATA enthält alle dynamisch modifizierbaren Daten (Registervereinbarungen) und ist beschreibbar.
Die TEXT-Sektion enthält alle gemeinsamen Zeichenketten, z.B. Bezeichnungen für die Steckeinheiten und Fehlerausschriften.
INITIALIZE wird zu Beginn jedes Durchlaufes automatisch vom Supervisor angesprungen.
Mit der CLEANUP Sektion wird jeder Programmdurchlauf abgeschlossen. Sie wird automatisch vom Supervisor aufgerufen.
SUMMARY erzeugt eine Zusammenfassunge aller Testergebnisse seit dem Laden des Programms.

RXBUF, TXBUF und TMPBUF sind Pufferbereiche für Datenübertragungen.

14.5.3. Programmlaufzeit

Das Laden des Programmes PVCBA dauert 10 sec, die Abarbeitung dauert 1 min.

14.5.4. Abarbeitungsbedingungen

Da das Diagnoseprogramm PVCBA vom Supervisor aus geladen wird, ist eine Speicherverlegung nicht möglich.

14.5.5. Flagverwendung

Entfällt.

14.5.6. Allgemeine Programmanforderungen

Entfällt.

14.5.7. Fehlerermittlung

Im Fehlerfall erscheint folgendes Protokoll:

```
********  ZZ - PVCBA - DWA40 REPAIR DIAGNOSTIC - 1.0  ********
 Pass 1, test y, subtest z, error v, aa-bbb-cccc dd:ee:ff:nn
 Hard error while testing DW0: MSGnr.
POSSIBLE BAD BOARD(S): ...
GOOD: ........
BAD:  ........
XOR:  ........

********  End of Hard error number  v  ********
```

y Testnummer
z Subtestnummer
v Fehlernummer
aa-bbb-cccc Tag Monat Jahr
dd:ee:ff:nn Stunden Minuten Sekunden Zehntelsekunden

Für MSGnr. können folgende Meldungen stehen:

MSG1	RECEIVED AN UNEXPECTED MACHINE CHECK!
MSG2	CONTENTS OF CNFGR REG. WRONG!
MSG3	CONTENTS OF DCR REG. WRONG!
MSG4	AN INTERRUPT SHOULD HAVE OCCURED
MSG5	SHOULD HAVE RECEIVED A MACHINE CHECK
MSG6	UNEXPECTED INTERRUPT!
MSG7	DATA COMPARE ERROR!
MSG8	PURGE FAILED!
MSG9	UASR CONTENTS BAD!
MSG10	DPR(N) CONTENTS BAD!
MSG11	BRRVR(N) CONTENTS BAD!
MSG12	SET CTL REG READ ERROR!

```
MSG13   FMER CONTENTS BAD!
MSG14   FUBAR CONTENTS BAD!
MSG15   IT DID NOT ENTER INTO ISR4 ROUTINE
MSG16   CONTENTION LOGIC PROBLEM
MSG17   RECEIVED AN ILLEGAL INTERRUPT
MSG19   SSYNTO BIT NOT SET WITHIN 16US
MSG20   UBIC BIT IN CNFGR DID NOT GET SET
MSG21   MICROSEQUENCER IDLE BIT IN DCR NOT CLEAR
MSG22   NOTE - "SET" IS NOT IN THE SYSTEM HENCE NOT TESTING IT !
MSG24   CONT. OF UACR REG. WRONG!
MSG25   CONT. OF MAP REG. WRONG!
MSG26   CONT. OF BRSVR REG. WRONG!
MSG27   UBIC BIT IN CNFGR S/B CLEAR
MSG28   UBINIT BIT IN CNFGR WAS SET PREMATURELY
MSG28A  UBINIT BIT IN CNFGR WAS CLEARD PREMATURELY
MSG29   UBINIT BIT IN CNFGR S/B SET
MSG30   UBIC BIT IN CNFGR S/B SET NOW
MSG31   PDN BIT IN CNFGR S/B CLEAR NOW
MSG32   CONT. OF DATA PATH REG. WRONG!
```

Durch die ausgedruckte Fehlermeldung ist es leicht möglich, die zugehörige Stelle im Programm zu finden. Zusätzlich gibt es eine weitere Fehlerausschrift, die ohne SET, aber bei aufgetrennter NPG-Leitung wirksam wird und zum vorzeitigen Programmabbruch führt: SKRBUS FAULT - SKRBUS SELECT TIME OUT

14.5.8. Programmitteilungen

Wenn das Testprogramm ohne Fehler abgearbeitet wird, so enthält das Protokoll eine Auflistung der Tests (siehe Punkt 14.8.) und danach:

```
** End of run, 0 errors detected, pass count is 1,
   time is aa-bbb-cccc dd:ee:ff:nn
DS>
```

Bei fehlender SET erfolgt vor der Auflistung der Tests die Mitteilung:

```
NOTE - "SET" IS NOT IN THE SYSTEM   HENCE NOT TESTING IT
```

14.5.9. Leistungsparameter des Testobjekts

Entfällt.

14.5.10. Aktionen bei unerwarteten Fehlern

Bei Netzausfall oder unerwarteten Interrupts kann das Diagnoseprogramm neu geladen werden, wobei die Abarbeitung vollständig oder ab einem bestimmten Test durchgeführt werden kann (diese Möglichkeiten bietet der Supervisor).

14.5.11. Testfolgen bei mehreren Einheiten

Entfällt.

14.5.12. Einstellungen für Ablauf unter AFT

Entfällt.

14.5.13. Testsektionen

In diesem Programm gibt es nur die Standardsektion.

14.5.14. Verschiedenes

Unter Maschinenprüfung ist in diesem Programm folgendes zu verstehen: Die Maschinenprüfung ist eine Service-Routine, in der das Steuerregister UACR des SBA gelöscht wird und nach dem Austesten des CPU-Typs die Löschung der Prozessorregister 30%X SBIFS oder 4A%X EHSR erfolgt. Die Prozessorregister werden gelöscht, um im weiteren Verlauf doppelte Fehler-Halts zu verhindern.

Unter Purge ist eine software-initiierte Funktion des SKRBUS-Adapters zu verstehen, während der der BDP gelöscht und initialisiert wird (Reinigungsfunktion).

14.6. Testbeschreibungen

14.6.1. Test 1 - CNFGR-Registerprüfung

CNFGR-Registers = Configuration-Register
Dieser Test prüft, ob beim Schreiben des CNFGR-Registers keine Maschinenprüfung (siehe 14.5.14.) erfolgt, ob die erwarteten Bits gesetzt werden, löscht und überprüft das Löschen und liest/schreibt eine 1 auf die Löschbits.

Testschritte:
Beachte: Vor dem Start des Programms muß der Operator das Kommando UNJAM eingeben, dann erfolgt die Initialisierung von der Konsole.
Zuerst wird das CNFGR-Register gelesen, wobei bei dem Leseversuch keine Maschinenprüfung erfolgen dürfte.
CNFGR <0:7> sollte den SKRBUS-Adaptercode enthalten.
CNFGR <16> sollte gesetzt sein.
Bit <22> (Adapter-POWER-UP) wird ignoriert.
Schreiben einer 1 auf <16> (SKRBUS-INIT-COMPLETE).
Lesen von CNFGR <16> — sollte gelöscht sein.

14.6.2. Test 2 - DCR-Register-Test

DCR = Diagnostic-Control-Register
Es wird überprüft, daß bei der Adressierung des DCR-Registers keine Maschinenprüfung (siehe 14.5.14.) erfolgt und daß die erwarteten Bits gesetzt und gelöscht werden.

Testschritte:
Lesen des DCR-Registers, wobei keine Maschinenprüfung erfolgen dürfte.

Bits <28,29,30,31> (DDPP=Defeat-Data-Path-Parity, DMP=Defeat-Map-Parity, DINTR=Disable-Interrupt und SPARE) sollten Null sein.
Bit <27> (MICOK=Microsequencer O.K.) sollte gesetzt sein. Die anderen Bits werden nicht beachtet.
Schreiben von Einsen auf DCR <28,29,30,31>.Prüfen, ob die Bits gesetzt sind.
Löschen des DCR-Registers. Überprüfen, ob die Bits <28,29,30,31> gelöscht sind. Bit 27 ist noch gesetzt.

14.6.3. Test 3 - Test DCR und CNFGR

In diesem Test wird überprüft, ob beim Schreiben des DCR-Registers das CNFGR-Register unbeeinflußt bleibt.

Testschritte:
Beschreiben des DCR-Registers mit Einsen.
Lesen des CNFGR-Registers, wobei die Bits <0:7,16,22,23> nicht beachtet werden. Keine anderen Bits dürften mit Einsen beschrieben worden sein.
Löschen des DCR-Registers.

14.6.4. Test 4 - Test UACR-Register

UACR = Control-Register
Prüfung des IB Bus und der Befehlsdecodierung für die Steckeinheiten USI und UAI. Bei der Adressierung des UACR-Registers darf keine Maschinenprüfung (siehe 14.5.14.) erfolgen.
Außerdem wird getestet, ob alle R/W-Bits gesetzt und gelöscht werden. Nach der Initialisierung des SBA muß das UACR-Register gelöscht sein. Er prüft außerdem, daß der SBA innerhalb von 30 ms nach dem Setzen des ADINIT-Bits (Adapter Init <0>) initialisiert wird.

Testschritte:
Schreiben einer 1 nach UACR <0>, dürfte keine Maschinenprüfung erzeugen.
40 ms warten.
Prüfen UACR - sollte Null sein.
Schreiben von Einsen auf UACR, außer auf <0,1>.
Lesen der Bits <2:7,26:30> (CNFIE, SUEIE, USEIE, BRIE, IFS und MRD) - sollten gesetzt sein.
Löschen von UACR und Überprüfung.
Jetzt wird überprüft, ob das UBIC-Bit in CNFGR innerhalb von 30ms nach dem Setzen des ADINIT-Bits in UACR gesetzt wird.

14.6.5. Test 5 - Test UASR-Register

UASR = Status-Register
Bei diesem Test wird kontrolliert, daß bei der Adressierung des UASR keine Maschinenprüfung (siehe 14.5.14) erfolgt und nach der Initialisierung des SBA das UASR-Register gelöscht ist.

Testschritte:
Initialisierung des SBA durch Setzen von UACR <0> (ADINIT).
Lesen von UASR - sollte Null sein, wobei keine Maschinenprüfung veranlaßt werden soll.

14.6.6. Test 6 - Initialisierungstest

Initialisierungstest durch Setzen von ADINIT.
Nach dem Setzen von ADINIT müssen die Register CNFGR, UACR, UASR und DCR ihren Anfangszustand haben.

Testschritte:
Setzen von UACR <0>.
Lesen von UACR - sollte gelöscht sein.
Lesen des CNFGR-Registers (SKRBUS-INIT-COMPLETE) <16> - sollte gesetzt sein und die Bits <0:7> sollten den Adaptercode enthalten.
Lesen von DCR - sollte gelöscht sein.
Lesen von UASR - solllte gelöscht sein.

14.6.7. Test 7 - SKRBUS-Netzausfall-Test

In diesem Test wird unter Steuerung des Programms die korrekte Folge von Operationen bei Stromausfall und beim Einschalten getestet. Durch Steuerung des UPF-Bits (SKRBUS-Power-Fail <1>) im UACR-Register werden die Bits UBPDN (SKRBUS-Power-down <17>), UBINIT (SKRBUS-Init-Asserted <18>) und UBIC (SKRBUS-Initialization-Complete <16>) im CNFGR-Register überwacht.

Testschritte:

Subtest 1 —— Setzen des UPF-Bits in UACR (SKRBUS-Power-Fail).
Löschen des UPF-Bits.
Lesen des UBPDN-Bits in CNFGR - sollte gesetzt sein.
Lesen des UBIC-Bits in CNFGR - sollte gelöscht sein.
Etwa 2ms warten - innerhalb dieser Zeit dürfte UBINIT nicht gesetzt werden.
Danach Warten von weiteren 5ms - bis zu der Zeit sollte UBINIT gesetzt sein.
Warten weiterer 20ms - UBIC müßte dann, infolge des SKRBUS-Einschalt-Prozesses gesetzt sein, wobei UPF (SKRBUS-Power-Fail) gelöscht wird.
Zusätzlich sollten UBINIT und UBPDN gelöscht sein.

Subtest 2 —— Initialisierung des SBA.
Setzen des UPF-Bits in UACR.
Lesen des UPDN-Bits, das gesetzt sein sollte und Lesen des UBIC-Bits, das gelöscht sein sollte.
Das UBINIT sollte 5ms lang gelöscht sein - dann sollte es gesetzt sein. Weitere 10ms warten und dann Prüfung, ob PDN und UBINIT noch gesetzt sind und ob UBIC noch gelöscht ist.
Löschen von UPF.
15ms warten - innerhalb dieser Zeit sollte das UBIC-Bit gesetzt sein.
UBPDN und UBINIT sollten jetzt gelöscht sein.

Subtest 3 —— Initialisieren des SBA.
Setzen aller 64%D Vektoren mit einer illegalen Interrupt-Service-Routinen-Adresse, um unerlaubte Interrupts zu finden. Jetzt wird der SBA-Vektor auf eine zuständige Service-Routinen-Adresse eingestellt, das UPF-Bit in UACR gesetzt und 10ms auf das Beenden des SKRBUS- Abschaltens gewartet.
Setzen von UACR <2> (CNFIE). Verringern von BR (kleiner als BP4) - ein Interrupt sollte auf diesem Adapter-Niveau erfolgen.

14.6.8. Test 8 - ADINIT Taktierung und Netzausfall.

Test 8 überprüft, ob nach Wirksamwerden von ADINIT (Adapter-Init) der Zustand des Mikro-Sequenzers in Ordnung ist und die Netzausfallbehandlung richtig erfolgt.

Voraussetzungen:
Die Bits UBPDN, UBIC, UBINIT und die entsprechenden Monoflops arbeiten im Stromausfallbehandlungstest.

Testschritte:
Initialisierung des SBA. Warten bis UBIC in CNFGR gesetzt ist.
Setzen des ADINIT-Bit in UACR.
Prüfen, daß UBPDN in CNFGR gesetzt ist und UBINIT in CNFGR gelöscht ist.
Das Micro-Sequencer-State-Bit <27> in DCR ist gelöscht und wird nach etwa 500us gesetzt werden.
Dann wird geprüft, ob das UBINIT-Bit in CNFGR nach etwa 2ms gelöscht ist und UBINIT innerhalb weiterer 5ms gesetzt wird. Nochmaliges Warten von 20ms - inzwischen müßte das UBIC-Bit in CNFGR gesetzt sein und das UBINIT-Bit sollte gelöscht sein.

14.6.9. Test 9 - Interrupt-Logik

Test der Konfiguration der Interrupt-Logik.
Nach ADINIT wird das UBIC-Bit <16> im CNFGR-Reg. gesetzt und ein Interrupt auf der Adapterstufe erzeugt.

Voraussetzungen:
Es kann auf die Register zugegriffen werden, UACR <2> (CNFIE=Configuration-Interrupt-Enable) ist in Ordnung und die Interrupt-Logik arbeitet teilweise.

Testschritte:
Initialisierung des SBA.
Prüfung, ob UBIC <16> in CNFGR gesetzt ist. Lesen aller BRRVR-Register diese müßten gelöscht sein.
Setzen UACR <2> (CNFIE). Verkleinern von BR auf BR3 - es müßte ein Interrupt erfolgen.
Verwenden der gleichen Interrupt-Service-Routine wie im vorangegangenen Test.
Nach der Rückkehr von der Routine wird UACR <2> (CNFIE) gelöscht.

14.6.10. Test 10 - Test SKRBUS-Mask-Error-Funktion.

Auf SKRBUS-Netzausfall und Schreiben auf eine SKRBUS-Adresse folgt eine Maschinenprüfung. Ebenso führt das Schreiben eines Langwortes auf eine SKRBUS-Adresse zu einer Maschinenprüfung.

Voraussetzungen:
SKRBUS-Netzausfall-Logik arbeitet teilweise.

Testschritte:
Setzen von UACR <1> (UPF).
Schreiben auf eine SKRBUS-Adresse - müsste eine Maschinenprüfung veranlassen.
Löschen von UACR <1> - 20ms warten.
Prüfen, ob CNFGR <17,18> auch gelöscht worden sind.
Danach Schreiben eines Langwortes auf eine SKRBUS-Adresse müßte eine Maschinenprüfung veranlassen.

Überprüfung von UASR <0> (SSYNTO) - sollte gesetzt sein.
Schreiben einer 1 auf UASR <0> um zu löschen.

14.6.11. Test 11 - Test SBA-Mask-Error-Funktion.

Das Schreiben eines Wortes auf ein SBA-Register führt zu einer Maschinenprüfung. Das Schreiben eines Langwortes nach den Registern FUBAR (Failed-SKRBUS-Adress-Reg.) und FMER (Failed-Map-Entry-Reg.) führt ebenso zu einer Maschinenprüfung (siehe auch 14.5.14).

Testschritte:
Initialisieren des SBA.
Schreiben eines Wortes auf das UACR-Register - müßte eine Maschinenprüfung veranlassen.
Dann wird ein Langwort auf FMER geschrieben - müßte eine Maschinenprüfung veranlassen.
Als nächstes wird ein Langwort auf FUBAR geschrieben - müßte ebenfalls eine Maschinenprüfung veranlassen.

14.6.12. Test 12 - Map-Register-Test

Lesen/Schreiben aller Map-Register.
Bei diesem Test wird geprüft, daß alle Map-Reg. adressiert werden können ohne eine Maschinenprüfung zu veranlassen. Alle Map-Reg. sind nach ADINIT Null. Durch das Schreiben und Lesen verschiedener Datenmuster können alle Bits von allen Map-Registern getestet werden.

Testschritte:
Initialisierung des SBA.
Lesen MR0:MR496 %D - müßten gelöscht sein.
Schreiben von Einsen auf MR0:MR496 %D - lesen und überprüfen.
Schreiben von Nullen auf MR0:MR496 %D - "
Schreiben von 1010 auf die Map-Register- "
Schreiben von 0101 auf die Map-Register- "
Schreiben von gleitenden Einsen auf die Map-Register - lesen und überprüfen
Schreiben von gleitenden Nullen auf die Map-Register - lesen und überprüfen
Nochmaliges Schreiben von Einsen auf MR(N); Initialisierung des SBA.
Dann Überprüfung, ob MR(N) gelöscht ist.
Für alle Map-Register wiederholen.

14.6.13. Test 13 - Map-Register-Test

Adresstest für alle Map-Register.
In diesem Test wird die eindeutige Adressierbarkeit aller Map-Register durch ein veränderliches Umkehrmuster überprüft. Die Map-Register sind aus 16 1k*1Bit RAM Chips aufgebaut.

Testschritte:
1) Adresse 0 (MR0) wird zuerst gelesen und bestätigt.
Dann wird ein Langwort 0000FFFF auf MR0 geschrieben, so daß der Speicherplatz 0 von allen 16 Chips (untere Hälfte von MR) alles Einsen enthält. Speicherplatz 0 (MR0) wird wieder gelesen um ihn zu überprüfen.
Dann wird FFFFFFFF auf MR0 geschrieben, so daß der Speicherplatz 1 von allen 16 Chips (obere Hälfte von MR) alles Einsen enthält.Dieser Speicherplatz wird wieder gelesen und überprüft.

Dieser Schritt 1 wird wiederholt bis alle Chip-Speicherplätze, d.h. alle Map-Register beschrieben und wieder gelesen sind. Der Speicher ist jetzt mit Einsen gefüllt.

2) Dann wird eine umgekehrte Operation ausgeführt.
Ein Langwort FFFF0000 wird auf MR495 geschrieben, so daß der Speicherplatz 1022%D (untere Hälfte von MR495) alles Nullen enthält. Er wird wieder gelesen und überprüft.
Dann wird das nächste niedrigere Map-Register (MR494) gelesen und überprüft, ob noch alles Einsen enthalten sind.
Alles Nullen (00000000) werden auf MR495 (obere Hälfte) geschrieben d.h. auf den Chip-Speicherplatz 1023%D.

3) Es wird das nächste kleinere Map-Register (MR494) gelesen und getestet, ob es noch alles Einsen enthält. Dieses Map-Register wird zuerst mit FFFF0000 und dann mit 00000000 beschrieben und, als Test, jedesmal gelesen.
Schritt 3 wird wiederholt bis MR0 wieder getestet ist. Der Speicher enthält jetzt alles Nullen.

4) Es werden verschiedene Adressfolgen kreiert.
Die Schritte 1-3 werden für MR1, MR3, MR5 ... MR495 ,
" MR1, MR5, MR9 ... MR493 ,
" MR1, MR9, MR17... MR489 ,
" ..... wiederholt,
bis zuletzt MR0 und MR256 erreicht wurden.

14.6.14. Test 14 - BRSVR-Register-Test

Lesen/Schreiben aller BRSVR-Register und Adresstest der BRSVRs (Buffer-Selection-Verification-Register).
Es wird getestet, ob auf alle BRSVR-Register ohne Maschinenprüfung zugegriffen werden kann. Alle Bits werden auf ständig 1, ständig 0 getestet und die BRSVR-Register werden auf duale Adressierbarkeit geprüft.

Testschritte:
Initialisierung des SBA.
Lesen aller BRSVR-Register - sollten gelöscht sein.
Schreiben von Einsen auf alle BRSVR - zur Überprüfung lesen.
Das Gleiche mit Nullen, 1010 und 0101 durchführen.
Anschließend Schreiben einer Eins (danach einer Null) auf alle BRSVR, die dann nach links verschoben wird. Es wird jeweils zur Überprüfung gelesen.
Danach werden verschiedene Daten auf BRSVR geschrieben und auf Richtigkeit, sowie eindeutige Adressierbarkeit geprüft.

14.6.15. Test 15 - SSYNTO-Bit-Test

Test SSYNTO-Bit (SKRBUS-Slave-Sync-Time-out <0>) in UASR.
Das Schreiben und Lesen auf und von SKRBUS-Adressen beeinflußt das SSYNTO-Bit.

Voraussetzungen:
Das UACR-Reg. läßt sich adressieren.

Testschritte:
Initialisierung des SBA.
Lesen des UASR-Registers - sollte gelöscht sein.
Schreiben auf einen nicht existierenden SKRBUS-Speicherplatz.

Lesen UASR Bit <0> (SSYNTO) - müßte gesetzt sein.
Jetzt erfogt eine Zeitüberprüfung - eine angenäherte Messung. Es wird getestet, ob nach dem Schreiben auf einen nicht existierenden Speicherplatz, das SSYNTO-Bit innerhalb von etwa 20us gesetzt wird.
Löschen UASR <0>.
Lesen von einem nicht existierenden SKRBUS-Speicherplatz.
Lesen UASR <0> (SSYNTO) - müßte gesetzt sein.
Löschen UASR <0>.

14.6.16. Test 16 - FUBAR-Register-Test

Test 16 überprüft, ob eine nicht existierende SKRBUS-Adresse in das FUBAR-Register abgespeichert wird. Außerdem wird das FUBAR-Register auf Selbsthaltung und Nicht-Selbsthaltung getestet.

Voraussetzungen:
UASR <0> (SSYNTO) ist in Ordnung.

Testschritte:
Subtest 1 --- Initialisierung des SBA.
Schreiben von 2 Wörtern auf eine nicht existierende SKRBUS-Adresse, beginnend ab Adresse 00. Lesen von FUBAR - sollte die SKRBUS-Adresse, die zuletzt eingeschrieben wurde, enthalten, aber um 2 Positionen nach rechts verschoben - Bit <17:2>.
Inkrementieren der SKRBUS-Adresse um 2 nach jedem Schreiben eines Wortes und Löschen von UASR <0> (SSYNTO).
Fortsetzen des obigen Ablaufes - Schreiben eines Wortes und Lesen von FUBAR, bis die SKRBUS-Adresse = 40000%X.
Beim ersten und zweiten Lesen von FUBAR sollte dieses Null enthalten, dann 1, 2, 3 usw.
Nach dem letzten Schreiben eines Wortes auf die nicht existierende SKRBUS-Adresse wird wiederUASR <0> geprüft - das gesetzt sein sollte.
Setzen von UACR <3> (SUEIE).
Verkleinern von BR und Test, ob ein Interrupt erfolgt.

Subtest 2 --- Prüfung auf Selbsthaltung des FUBAR-Registers.
Lesen von einer nicht existierenden SKRBUS-Adresse.
Warten bis das SSYNTO-Bit in UASR gesetzt worden ist.
Prüfen, ob FUBAR <17:2> die SKRBUS-Adresse enthält.
Das SSYNTO-Bit nicht löschen. Lesen von einer anderen SKRBUS-Adresse.
Warten bis das SSYNTO-Bit gesetzt wird. Prüfen, ob FUBAR <17:2> noch die alte SKRBUS-Adresse enthält, d.h. SSYNTO wird in FUBAR für den zweiten Lesevorgang gespeichert.
Dann wird das SSYNTO-Bit gelöscht, um das Nicht-Selbsthalten zu testen. Lesen von der gleichen SKRBUS-Adresse, wie beim letzten Mal. FUBAR müßte die letzte SKRBUS-Adresse <17:2> enthalten.

14.6.17. Test 17 - FMER-Register-Test

Test FMER-Register und IVMR-Bit (Invalid-Map-Register <4>) in UASR.
Wenn ein Map-Reg. verwendet wird, das nicht validiert ist, so wird die Map-Nummer in FMER gespeichert und in UASR wird das IVMR-Bit gesetzt.

Testschritte:
Initialisierung des SBA.
Lesen des UASR - sollte gelöscht sein.

Einstellen der Map-Register MR0:MR1EF auf eine ISB-Seite.
Auswählen von BDP1 (Buffered-Data-Path), das BYTE-OFFSET-Bit löschen.
Ungültig machen (Setzen) von <31> (MV) des Map-Registers, das benutzt wird, um alle anderen Map-Register gültig zu machen.
Schreiben eines Wortes auf die SKRBUS-Adresse 0.
Lesen von UASR <0,4> (SSYNTO,IVMR) - sollte gesetzt sein.
Lesen des FMER-Registers - sollte gelöscht sein (SKRBUS-Adresse 0).
Setzen UACR <4> (USEIE). Verkleinern von BR auf BR3 und Test, ob in der Adapterstufe ein Interrupt erfolgt.
Löschen UASR <0>, UACR <4>.
Nach der Rückkehr von der Interrupt-Service-Routine erfolgt das Schreiben eines Wortes auf die SKRBUS-Adresse 200%X unter Verwendung von MR1.
Lesen von FMER - sollte 1 sein.
Fortsetzen des Schreibens auf die SKRBUS-Adresse, so daß MR2:MR1EF%X verwendet werden und Prüfung von FMER, ob das richtige Map-Register, das verwendet wurde, enthalten ist.

14.6.18. Test 18 - DPR-Register-Test

Zugriffsmöglichkeit zu den DPR-Registern (Data-Path-Register).
Dieser Test überprüft, ob auf alle DPR-Reg. ohne Maschinenprüfung (siehe 14.5.14) zugegriffen werden kann. Die Register sind durch die Initialisierung gelöscht. Außerdem wird überprüft, daß der Mikro-Code DPR0 findet.

Testschritte:
Initialisierung des SBA.
Lesen DPR0:DPR15 - Inhalt sollte Null sein.
Schreiben von AAAAAAAA nach BRSVR4:BRSVR7.
Jedes Mal wird DPR0 gelesen um zu prüfen, ob der Mikro-Code DPR0 findet - der Inhalt von DPR0 sollte Null sein.

14.6.19. Test 19 - Map-Register-Disable-Funktion

Eine Gruppe von Map-Registern wird blockiert (unwirksam gemacht), und bei ihrer Verwendung werden SSYNTO und IVMR auf Fehler getestet, und durch die Verwendung der nächsten Gruppe, die nicht blockiert ist, wird geprüft, ob alle arbeiten.

Voraussetzungen:
Die Map-Register, UACR und UASR arbeiten.

Testschritte:
Initialisierung des SBA.
Lesen UACR <30:26> (Disable-Bits) - sollten gelöscht sein.
Laden von MR15%D so, daß es auf den ISB-Speicher zeigt; Verwenden von DP0.
Ausführen eines Datentransfers (Schreiben) vom ISB auf eine SKRBUS-Adresse entsprechend MR15.
Prüfen, ob UASR <0,4> (SSYNTO, IVMR) gelöscht sind.
Anschließend wird UACR <2> gesetzt - MR0:MR15%D werden unwirksam gemacht.
Wiederholen des Transfers mit MR15.
Lesen von UASR <0> (SSYNTO) - dieses Bit müßte auf Null gesetzt worden sein.
Wiederholen des Transfers mit MR16 - UASR <0> dürfte nicht gesetzt worden sein.
Dann wird UACR <30:26>=00010 gesetzt und die Map-Register MR31 und MR32 werden verwendet. Dabei wird überprüft, ob bei Verwendung von MR31 das SSYNTO-Bit gesetzt wird und bei Verwendung von MR32 nicht gesetzt wird.
Wiederholen des obigen Prozesses bis UACR <30:26>=11111 und MR494 und MR495

verwendet wurden.

14.6.20. Test 20 - Status-Bit-Test

Test der Status-Bits in den DPR-Registern ohne ISB-Schreiben (Interner-Synchron-Bus).
In den DPR1:DPR15 weden die Status-Bits auf Haftfehler für Logikpegel 1 und 0 getestet, ohne ein ISB-Schreiben.

Testschritte:
Initialisierung des SBA.

Subtest 1 --- MR0 als Zeiger auf einen Puffer im ISB (RXBUF) einstellen, Verwenden von BDP1.
Gleichzeitiges Schreiben eines Bytes auf RXBUF über den SKRBUS (W/A-Modus) und Prüfen, ob die Mask-Bits, SKRBUS-Adress-Bits, BNE-, BTE- und TF-Bits in Ordnung sind.
Erstes Byte-Schreiben - Mask-Bit0 (Bit16) ist gesetzt.
Zweites Byte-Schreiben - Mask-Bit0 + Mask-Bit1 (Bit16 + Bit17) sind gesetzt.
usw.
Schreiben von 7 Bytes. Wird für BDP1:BDP15 ausgeführt.
Es wird kein Purge (siehe 14.5.14) durchgeführt.

Subtest 2 ---Die Mask-Bits werden durch eine laufende Eins verändert.
Initialisieren des SBA. MR0 wie oben einstellen.
Schreiben eines Bytes nach RXBUF über SKRBUS und Prüfen, ob Mask-Bit0 gesetzt ist. Löschen von DPR1 durch Initialisieren von SBA.
Inkrementieren der SKRBUS-Adresse. Schreiben eines anderen Bytes. Mask-Bit1 sollte gesetzt sein - die anderen sollten gelöscht sein.
Schreiben von 7 solchen Bytes und Prüfen aller Mask-Bits (7). Ausführen dieser Operation für DPR1:DPR15.

Subtest 3 --- Jetzt wird ein Paar von laufenden Einsen verwendet.
Initialisierung des SBA. MR0 wieder wie oben einstellen (Subtest 1).
Schreiben eines Wortes nach RXBUF über den SKRBUS.
Prüfen, ob Mask0 + Mask1 (Bit16 + Bit17) gesetzt sind.
Löschen von DPR1 durch UBINIT. SKRBUS-Adresse um 2 erhöhen.
Schreiben eines anderen Wortes und Prüfen, ob die Bits Mask2 + Mask3 (Bit18 + Bit19) gesetzt sind.
Nochmaliges Löschen von DPR1, Addieren von 2 zu der SKRBUS-Adresse.
Danach Schreiben eines weiteren Wortes — Mask4 + Mask5 sollten gesetzt sein.

In den 3 Subtests wurde geprüft, ob die Bits BNE und TF gesetzt wurden, ob BTE gelöscht wurde und die richtige SKRBUS-Adresse in DPR stand.

14.6.21. Test 21 - Datentransfer-Test /Schreiben.

Dieser Test ist der Anfang der Daten-Transfer-Tests. Es wird ein Byte, dann ein Wort über DDP, BDP1:BDP15 aus einem Puffer im ISB-Speicher auf einen anderen Puffer in ISB über den SKRBUS unter Verwendung der Map-Register geschrieben. Dann prüft der Daten-Empfangs-Puffer den Status von DPR und UASR. Es wird auch das Schreiben von Daten mit gesetztem Byte-Offset-Bit getestet.

Voraussetzungen:
Alle Map-Register sind in Ordnung, aber sie sind noch nicht für Mapping verwendet worden. UASR ist in Ordnung.

Testschritte:
Initialisierung des SBA.
Setzen von MR1 auf RXBUF-Seite (ISB-Speicher).
Byte-Offset-Bit löschen. MRV <31> (Map-Register-Valid) setzen.
RXBUF beginnt an einer Langwortgrenze.

A) Schreiben eines Bytes von TXBUF auf die SKRBUS-Adresse entsprechend MR1.
Prüfen, ob UASR=0 und Prüfen der Daten in RXBUF.

B) Wiederholen von A, aber mit dem Schreiben eines Wortes.

C) Laden von MR1 mit dem BDP1-Zeiger.
Schreiben eines Bytes.
Lesen von DPR(N) <BTE, BTF, BNE, Mask-Bit1, SKRBUS-Adress-Bits> und danach Ausführung eines Purge (siehe 14.5.14).
Prüfen, ob UASR=0, Prüfen der Daten, BNE=0.
Wiederholen von C für BDP2:BDP15.

D) Wiederholen von C, aber mit gesetztem Byte-Offset-Bit. Das Mask-Bit2 wird nur in DPR(N) gesetzt sein.

E) Schreiben eines Wortes von TXBUF nach RXBUF über den SKRBUS. Verwenden von BDP1.
Lesen von DPR(N) <BTE, BTF, BNE, Mask-Bit1 & 2, SKRBUS-Adress-Bits>.
Purge (siehe 14.5.14). Prüfen von UASR=0 und Prüfen der Daten.
Wiederholen von E mit BDP2:BDP15.
Wiederholen von E mit gesetztem Byte-Offset-Bit.

14.6.22. Test 22 - Test Purge doing a NOP

Test von Purge (siehe 14.5.14) beim Schreiben von 4 Wörtern.

Voraussetzungen:
Map-Reg.40%X, BDP1 und Purge arbeiten.

Testschritte:
MR40%X wird als Zeiger auf RXBUF (ISB-Speicher) eingestellt.
Verwendet BDP1.
Laden von 4 Langwörtern auf RXBUF.
Schreiben von 4 Wörtern von TXBUF (enthält AAAA) nach RXBUF über den SKRBUS.
Prüfen der Daten in RXBUF.
Prüfen, ob das 3. Langwort in RXBUF unverändert ist.
Prüfen des BNE-Bits in DPR1 - sollte gelöscht sein.
Dann Ausführen von Purge (siehe 14.5.14).
Nochmaliges Prüfen von RXBUF, um sicher zu machen, daß das 3. Langwort noch unverändert ist. TXBUF und RXBUF beginnen an einer Langwortgrenzen. Nach dem 4. Worttransfer findet das Purge automatisch vor dem Purge-Kommando statt.

14.6.23. Test 23 - DPR-Mask-Bit-Test

Test der DPR-Mask-Bits und der empfangenen Daten — 8 Bytes.
Es erfolgt ein Schreiben von 8 Bytes über BDP1:BDP15 und ein Prüfen von DPR(N) und der empfangenen Daten.

Testschritte:
MR0 als Zeiger auf RXBUF einstellen. Über den SKRBUS (unter Verwendung von

BDP1:BDP15) wird ein Byte geschrieben und die Bits BNE, BTE, BTF, Mask-Bits und die SKRBUS-Adress-Bits in DPR(N) werden geprüft.
Diese Operation wird für das Schreiben von 8 Bytes ausgeführt. Dieses Schreiben wird unter Verwendung von BDP1:BDP15 15 mal ausgeführt und jedes Mal wird DPR(N) geprüft.
Ebenso werden die Daten in RXBUF und TXBUF bei jeder Verwendung von BDP verglichen.

14.6.24. Test 24 - Adressleitungstest

Test auf Kurzschlüsse oder Unterbrechungen in den Adressleitungen zu den (Mask) RAM. Außerdem wird getestet, ob die RAM arbeiten.

Voraussetzungen:
Map-Reg. 40%X, BDP1 arbeiten.

Testschritte:
MR40%X wird als Zeiger auf eine ISB-Speicherzelle eingestellt.
Schreiben eines Bytes über BDP1 zum SKRBUS.
Schreiben von 2 Bytes über BDP2,
von 3 Bytes über BDP3
... usw.
bis 7 Bytes über BDP7;
und wieder ein Byte über BDP8
bis BDP15 verwendet wurde.

Lesen von DPR1:DPR15 und Prüfen, ob die richtigen Mask-Bits, BNE, BTF gesetzt sind, das BTE-Bit gelöscht ist und die SKRBUS-Adresse <17:2> in jedem DPR steht.

14.6.25. Test 25 - BDP-Adresstest

BDP-Adresstest (Buffered-Data-Path).
Es wird die duale Adressierung aller gepufferten Datenwege getestet.

Vorausstzungen:
Alle Map-Reg. sind in Ordnung. Normales W/A-Schreiben arbeitet. Die einzelnen BDP arbeiten.

Testschritte:
Laden von TXBUF mit 45%D verschiedenen Wörtern.
MR0:MR15 als Zeiger auf RXBUF im ISB-Speicher einstellen.
MR0 enthält den BDP1-Zeiger, MR1 den BDP2-Zeiger ... MR14 den BDP15-Zeiger.
Schreiben von 3 Wörtern von TXBUF nach RXBUF über jedes Map-Register, so daß jedes Mal ein anderer BDP verwendet wird.
Vergleich der empfangenen Daten in RXBUF mit den gesendeten Daten.

14.6.26. Test 26 - Test der BUSY-Bestätigungs-Funktion

Voraussetzungen:
MR 40%X, BDP1 arbeiten.

Testschritte:
MR40 wird für ein W/A-Schreiben nach RXBUF gesetzt. Verwendet wird BDP1.
Ausführung des Schreibvorganges und eines Purge (siehe 14.5.14).

Danach wird sofort DPR1 gelesen.
Lesen UASR <10> (Read-Data-Time-out) - sollte gelöscht sein, um nachzuweisen, daß die SBA BUSY-Bestätigung arbeitet.

14.6.27. Test 27 - SKRBUS-Schreiben

Prüfen des SKRBUS mit vielen verschiedenen Datenmustern. Es werden alle 4 Steckeinheiten geprüft.

Voraussetzungen:
MR 40%X, BDP1 und Purge (siehe 14.5.14) arbeiten.

Testschritte:
Ausführen vieler Schreiboperationen von TXBUF (ISB) zum SKRBUS, wobei der SKRBUS auf RXBUF (ISB) abgebildet wird. MR40%X und BDP1 werden verwendet.
TXBUF enthält verschiedene Datenmuster
- alles Einsen
- alles Nullen
- 1010....
- 0101....
- laufende Einsen
- laufende Nullen.

Prüfen der Daten in RXBUF.

14.6.28. Test 28 - Mikro-Controller-Test

In diesem Test wird überpüft ob der Mikro-Controller während des Schreibens auf verschiedenen Startadressen beginnt.

Voraussetzungen:
MR 40%X, BDP1 und Purge (siehe 14.5.14) arbeiten.

Testschritte:
MR40%X wird als Zeiger auf RXBUF eingestellt. BDP7 wird verwendet.
8maliges Schreiben von 16 Wörtern von TXBUF nach RXBUF über den SKRBUS, wobei jedes Mal der Zeiger nach RXBUF (ISB) um 2 erhöht wird. Jedes Mal werden die Daten in RXBUF überprüft.
16maliges Schreiben von TXBUF nach RXBUF über den SKRBUS, wobei jedes Mal der Zeiger nach RXBUF um 1 erhöht wird. Die Daten in RXBUF werden jedes Mal überprüft.

14.6.29. Test 29 - Mikro-Controller-Test

Test zur Prüfung von Startadressen während des Schreibens mit gesetztem Byte-Offset-Bit.
Es wird getestet, ob der Mikro-Controller beim Schreiben auf verschiedenen Startadressen beginnt, wobei im Map-Register das Byte-Offset-Bit (BO) <25> gesetzt ist.

Voraussetzungen:
MP 40%X, BDP1 und Purge (siehe 14.5.14) arbeiten. Ebenso arbeitet die Byte-Offset-Funktion beim Start von Adresse 0 (normal).

Testschritte:
MR40%X als Zeiger nach RXBUF einstellen. BDP8 wird verwendet. Setzen des

Byte-Offset-Bit.

Subtest 1 --- 8maliges Schreiben von 16 Wörtern von TXBUF nach RXBUF über den SKRBUS - jedes Mal Erhöhen des Zeigers nach RXBUF um 2. Überprüfen der Daten in RXBUF.
Nach dem Datenvergleich in RXBUF wird der RXBUF-Zeiger um 1 erhöht - wegen dem Byte-Offset.

Subtest 2 --- 16maliges Schreiben von 32 Bytes von TXBUF nach RXBUF über den SKRBUS - jedes Mal wird der Zeiger nach RXBUF um 2 erhöht.
Überprüfen der Daten in RXBUF.
Nach dem Datenvergleich in RXBUF wird der RXBUF-Zeiger wegen dem Byte-Offset um 1 erhöht.

14.6.30. Test 30 - Datenübertragungstest - Lesen.

Hierbei handelt es sich um ein Lesen im W/A-Modus.Es wird ein Wort vom ISB zum ISB über den SKRBUS über DDP, BDP1:BDP15 gelesen. Dann werden 4 Wörter gelesen. Danach wird alles, mit im Map-Reg. gesetztem Byte-Offset-Bit, wiederholt.

Voraussetzungen:
MR 40%X, DDP, BDP1:BDP15 und Purge (siehe 14.5.14) arbeiten während des Schreibens.

Testschritte:
MR40 als Zeiger auf TXBUF in ISB einstellen. Verwendet DP0.

Subtest 1 --- Lesen eines Wortes von TXBUF nach RXBUF.
Prüfen des Datenwertes und Test, ob UASR=0 ist.

Subtest 2 --- Lesen eines Wortes von TXBUF nach RXBUF.
Verwenden von BDP1:BDP15. Anschließend Ausführen von Purge.
Prüfen von DPR(N) (BTE, BTF, BNE).

Subtest 3 --- Lesen von 4 Wörtern von TXBUF (Quad-Word-aligned d.h. an einer Quadwort-Grenze beginnend) nach RXBUF über den SKRBUS.
Nach jedem Lesen werden DPR(N) und UASR gelesen.
Es wird getestet, ob nach dem Lesen des 4. Wortes das BNE-Bit gesetzt ist, um zu bestätigen, daß sich ein Prefetch ereignet hat (vorausschauendes Laden).
Überprüfen der Daten in RXBUF.

Subtest 4 --- Wiederholen von Subtest 3, aber mit gesetztem Byte-Offset-Bit.

14.6.31. Test 31 - Mikro-Controller-Test

Der Test überprüft den Start des Mikro-Controllers beim Lesevorgang von verschiedenen Startadressen.

Voraussetzungen:
MR 40%X, UASR arbeiten.
Der Mikro Controller arbeitet, wenn er ab Adresse Null gestartet wird (normal).

Testschritte:
MR40 als Zeiger auf TXBUF setzen.
BDP2 wird verwendet. Laden von TXBUF mit bekannten Daten.

Subtest 1 --- 8maliges Lesen von TXBUF nach RXBUF, wobei jedes Mal der TXBUF-Zeiger um 2 erhöht wird.
Überprüfen der Daten in RXBUF. Nach jedem Lesen wird TXBUF mit neuen Daten geladen.

Subtest 2 --- Wiederholen des obigen Tests mit gesetztem Byte-Offset-Bit. Beim Vergleich der Daten in RXBUF wird wegen dem Byte-Offset eine Eins zu dem TXBUF-Zeiger addiert.

14.6.32. Test 32 - Map-Register-Test

Map-Grenzen-Überschreitung während des Schreibens.
Es werden 2 Seiten (512%D Wörter) + 1 Wort-Datenwert vom ISB zum ISB über den SKRBUSgeschrieben. Dabei sind die Map-Register MR(N) und MR(N+1) gültig aber MR(N+2) ist nicht gültig. Dann wird geprüft, daß 512%D Wörter geschrieben worden sind, aber das 513. Wort nicht. Dieser Test wird für alle 496%D Map-Register ausgeführt.

Voraussetzungen:
Alle Map-Reg., BDP1, UASR und Purge (siehe 14.5.14) arbeiten — aber nicht alle Map-Reg. sind bis jetzt auf Funktionsfähigkeit geprüft.

Testschritte:
Laden von TXBUF mit 513%D Datenwörtern - 1,2,3,4,... 513.
Löschen von RXBUF und von allen Map-Registern.
Validieren von MR(N) und MR(N+1); BDP1 wird verwendet.
Laden von MR(N) mit RXBUFPG# (Seitenadresse) und von MR(N+1) mit RXBUFPG#+1.
Schreiben von 513%D Wörtern von TXBUF nach RXBUF über den SKRBUS.
Prüfen, ob 512 Datenwörter in RXBUF mit TXBUF übereinstimmen. Das 513. Wort in TXBUF sollte nicht mit RXBUF übereinstimmen.
UASR <0,4> (SSYNTO,IVMR) sollten gesetzt sein.
Löschen von SSYNTO, IVMR und MR(N) (zuerst das alte Map-Register).
Danach wird die gleiche Prozedur mit MR(N+1) und MR(N+2) solange wiederholt, bis alle 496 Map-Register in den Test einbezogen waren.
Lesen von FMER nach jedem Durchlauf - müßte MR(N+2) enthalten.

14.6.33. Test 33 - Map-Register-Test

Map-Grenzen-Überschreitung während des Lesens.
Hierbei werden 2 Seiten (512%D Wörter) + 1 Wort-Datenwert vom ISB zum ISB über den SKRBUS gelesen. Dabei sind die Map-Reg. MR(N) und MR(N+1) gültig, aber MR(N+2) ist nicht gültig. Dann wird geprüft, ob die 512 Wörter gelesen worden sind, aber das 513. Wort nicht. Das wird für alle 496%D Map-Reg. wiederholt.

Voraussetzungen:
Alle Map-Reg., BDP1, UASR und Purge (siehe 14.5.14) arbeiten beim Schreiben.

Testschritte:
Laden von TXBUF mit 513%D Datenwörtern - 1,2,3,4, 513.
Löschen von RXBUF und aller Map-Register.
Validieren von MR(N), MR(N+1); Verwenden von BDP13%D.
Laden von MR(N) mit TXBUFPG# (Seitenadresse) und MR(N+1) mit TXBUFPG#+1.
Lesen der 513 Wörter von TXBUF nach RXBUF über den SKRBUS.
Prüfen, ob 512 Wörter (Daten) in RXBUF mit TXBUF übereinstimmen.
Das 513. Wort von TXBUF dürfte nicht nach RXBUF gelangt sein.
UASR <0,4> (SSYNTO,IVMR) sollten gesetzt sein - werden anschließend gelöscht.

Löschen der MR(N) (zuerst das alte Map-Register).
Danach wird die gleiche Prozedur mit MR(N+1) und MR(N+2) wiederholt, bis alle 496 Map-Register verwendet worden sind. FMER wird nach jedem Durchlauf gelesen - es sollte MR(N+2) enthalten.

14.6.34. Test 34 - Ein-/Ausgabe-DATO-Funktion

In diesem Test werden 2 Wörter im W/A-Modus nach DCR geschrieben (entspricht einer E/A-DATO-Funktion) und anschließend überprüft.

Voraussetzungen:
MR 40%X, BDP2 und DCR arbeiten.

Testschritte:
MR40%X wird als Zeiger auf DCR eingestellt. BDP2 wird verwendet.
Einschreiben in die obere Hälfte von DCR über den SKRBUS (entsprechend MR40).
Das Schreiben von 2 Wörtern ist notwendig. DCR wird sofort gelesen und überprüft, ob es korrekt beschrieben wurde.

14.6.35. Test 35 - E/A-DATO/B-Funktion

Durch das Schreiben von 4 Bytes nach DCR im W/A-Modus (entspricht einer E/A-DATO/B-Funktion) und anschließende Überprüfung wird diese Funktion getestet.

Voraussetzungen:
MR40%X, BDP2 und DCR arbeiten.

Testschritte:
Setzen von MR40%X als Zeiger auf das DCR-Register. Verwenden von BDP2. Schreiben eines Bytes in das obere Byte von DCR über den SKRBUS entsprechend MR40. Insgesamt müssen 4 Bytes geschrieben werden. Sofortiges Lesen und Überprüfen von DCR.

14.6.36. Test 36 - Test E/A-DATI (B01)

Bei diesem Test wird vom Speicher-Konfigurations-Register B im W/A-Modus ein zuvor gesetztes Bit <8> über SKRBUS gelesen und nachgeprüft, ob das entsprechende Bit gesetzt war.

Voraussetzungen:
MR 40%X, BDP2 arbeiten.

Testschritte:
Setzen eines Bits (Bit8) im Speicher-Konfigurations-Register B. MR40%X wird als Zeiger zu diesem Register eingestellt. BDP2 wird verwendet. Lesen von CNFGR über den SKRBUS entsprechend MR40. Prüfen, ob CNFGR B <8> gesetzt war.

14.6.37. Test 37 - Test E/A-DATI-(B23)-Funktion

Ein im Speicher-Konfigurations-Register C gesetztes Bit <30> wird im W/A-Modus über den SKRBUS gelesen. Anschließend wird überprüft, ob das Bit gesetzt war.

Voraussetzungen:
MR 40%X und BDP2 arbeiten.

Testschritte:
Setzen von Bit <30> im Speicher-Konfigurations-Register C (CNFGR C). MR40 wird als Zeiger auf CNFGR C eingestellt. BDP2 wird verwendet. Lesen von CNFGR C über den SKRBUS entsprechend MR(40). Bit <30> sollte gesetzt sein.

14.6.38. Test 38 - Interlock-Logik und DAT-I/P-Test

Die Ausführung eines BISW (Bit Set Word) auf ein Non-Quad-Word-aligned Speicherplatz (Speicherzelle, die nicht auf einer Quad-Word-Grenze beginnt) im W/A-Modus über DDP und deren Überprüfung sowie die Wiederholung über BDP und die Prüfung, die bei einer Fehlfunktion SSYNTO ergibt, testen die Interlock Logik und die DAT-I/P-Funktion.

Testschritte:
MR40%X wird als Zeiger auf einen nicht auf einer Quadwort-Grenze beginnenden ISB-Speicherplatz eingestellt. Verwendet wird BDP1.
Ausführen eines BISW zu diesem Speicherplatz.
Prüfen von DPR1 - die richtigen Mask-Bits und BNE sollten gesetzt sein. Lesen von UASR <0> (SSYNTO) - sollte gesetzt sein - dies beweist, daß DAT-I/P über einen BDP ausbleibt.
DATO findet statt - Daten werden in BDP1 sein.
Ausführen eines Purge (siehe 14.5.14). Direktes Lesen eines Speicherplatzes - müßte den Datenwert enthalten.
Löschen von UASR <0> und Löschen des Speicherplatzes. Jetzt wird wieder ein BISW über den SKRBUS zu dem gleichen Speicherplatz ausgeführt, aber unter Verwendung von DP0. Lesen von UASR <0> (SSYNTO) - sollte gelöscht sein. Direktes Lesen des Speicherplatzes - sollte Daten enthalten - um nachzuweisen, daß die DAT-I/P-Funktion über DP0 arbeitet.

14.6.39. Test 39 - Test Befehls-Adressen Fehler

Beim Schreiben nur eines Wortes im W/A-Modus in das DCR-Register wird ein Purge (siehe 14.5.14) ausgeführt, wenn ein Command-Transmit-Error vorliegt (CXTER <7> imUASR-Register).

Testschritte:
MR40 als Zeiger auf DCR setzen, BDP2 wird verwendet.
Schreiben nur eines Wortes in DCR über eine SKRBUS-Adresse entsprechend MR40.
Lesen von UASR <7> (Command-Transmit -Error) - sollte gesetzt sein.
Lesen von DPR2 <30> (Buffer-Transfer-Error) - sollte gesetzt sein.
Setzen UACR <4> (USEIE) - es wird ein Interrupt in der Adapterstufe auftreten. Verwenden der gleichen Service-Routine wie zuvor. Löschen von UASR, UACR.

14.6.40. Test 40 - Test Befehls-Adress-Time-out-Fehler

Beim Schreiben eines Wortes auf eine nicht existierende ISB-Speicher-Seite ergibt sich ein Command-Transmit-Time-out (CXTO <6> in UASR).

Voraussetzungen:
MR 40%X, BDP1 und Schreiben auf vorhandenen Speicherplatz arbeiten.

Testschritte:
MR40%X als Zeiger auf eine nicht existierende ISB-Speicher-Seite einstellen. BDP1 wird verwendet. Schreiben eines Wortes auf einen Platz in dieser Seite

über die SKRBUS-Adresse entsprechend MR40.
Prüfen von UASR <6> (Command-Transmit-Time-out) - sollte gesetzt sein.

14.6.41. Test 41 - Lost-Error-Bit-Funktion

Dieser Test prüft, ob nach Verursachen einer Fehlerbedingung, wenn ein weiterer Fehler auftritt, das Lost-Error-Bit (LEB <2>) in UASR gesetzt wird und das 2. Fehlerbit nicht gesetzt wird.

Voraussetzungen:
MR 40%X, DDP, IVMR Error, SSYNTO ERR CATO, CXTER (Command-Transmit-Error <7> in UASR) arbeiten.
Es gibt 5 Subtests.
Subtest 1, Error 1 und 2 - Erzeugen eines CXTO-Error
" 2, " 1 " 2 - " " CXTER- "
" 3, " 1 " 2 - " " MRPF (Map-Register-Parity-Failure)
" 4, " 1 " 2 - " " DPPE (Data-Path-Parity-Error)
" 5, " 1 " 2 - " " erzeugt zuerst einen CXTO-Error und kreiert dann einen IVMR-Error.

Testschritte:
Zweck ist das Setzen von Fehlerbedingungen und Prüfen, ob die richtigen Fehlerbits in UASR gesetzt sind.
Danach wird ein anderer Fehler erzeugt und geprüft, ob das Fehlerbit infolge des 2. Fehlers nicht gesetzt wird und statt dessen das Lost-Error-Bit (LEB) gesetzt ist.

Subtest 1 --- Ausführung eines Wrap-around-Schreibens von TXBUF nach RXBUF unter Verwendung von MR40%X, wobei es nicht validiert wird.
Prüfen UASR <4> (IVMR) - sollte gesetzt sein.
Prüfen UASR <2> (LEB) - sollte gelöscht sein.
Dann wird ein RDS-Fehler (Read Data Substitute) erzeugt - Laden von 011 nach F020; Löschen F024; Unwirksam machen von Controller-Error-LOG ; Laden von 2FF nach 20002004. Dann wird F020 gelesen - es müßte ein Fehler auftreten.
Prüfen von UASR <9> (RDS= Read-Data-Substitute) - müßte noch gelöscht sein.
Prüfen von UASR <2> (LEB) - sollte gesetzt sein.
Schreiben einer Eins nach UASR <2>, um es zu löschen.
UASR <4> (IVMR) wird nicht gelöscht.

Subtest 2 --- Erzeugen eines CXTER (Command-Transmit-Error) Fehlers. Schreiben eines Wortes auf eine nicht existierende ISB-Speicherzelle im W/A Modus.
Prüfen UASR <6> (CXTO) - sollte gelöscht sein.
Prüfen UASR <2>(LEB) - sollte gesetzt sein.
Löschen von LEB.

Subtest 3 --- Ein Command-Transmit-Error wird erzeugt.
Schreiben nur eines Wortes in das DCR-Register über den SKRBUS unter Verwendung von MR40. Prüfen von UASR <7> (Command-Transmit-Error) - sollte gelöscht sein. Löschen des LEB-Bits.

Subtest 4 --- Erzeugen eines Data-Path-Parity-Error.
Laden eines Datenwortes, das eine gerade Anzahl von Einsen enthält, nach TXBUF. Setzen des Defeat-Data-Path-Parity-Bits <28> im DCR-Register. Schreiben eines Bytes nach RXBUF im W/A-Modus, 10us warten. Test des Data-Path-Parity-Error-Bits <5> im Statusregister - müßte Null sein.

Subtest 5 --- Erzeugen eines CXTO-Error und anschließend eines IVMR-Error.

Prüfen ob IVMR-Bit nicht gesetzt ist und statt dessen LEB gesetzt ist.
Schreiben eines Wortes über den SKRBUS auf eine nicht existierende ISB-Speicheradresse, 20us warten. Prozessorprioritätsstufe auf BR3 verkleinern.
Test ob CXTO-Error-Bit <6> in UASR gesetzt ist - sollte gesetzt sein.
Schreiben eines Wortes nach RXBUF über den SKRBUS.
Test IVMR <4> in UASR (Invalid-Map-Register-Bit) - sollte Null sein. Test des Lost-Errror-Bit.

14.6.42. Test 42 - DEF.-, MAP.-, PAR.-Funktion

Test des SBA, der den ISB unter Verwendung der DEF.-, MAP.- und PAR.-Funktionen unterbricht.
Dieser Test prüft, ob bei der Ausführung eines Lesevorganges im W/A-Modus, bei dem ein Map-Register verwendet wird, das eine gerade Zahl von Einsen in irgendeinem Byte der oberen und unteren Hälfte enthält, das Defeat-Map-Parity in DCR aufgehoben wird und ein Interrupt erfolgt.

Voraussetzungen:
MR 40%X, DDP und UASR sind in Ordnung.

Testschritte:
Verwendung von DDP --
Setzen von MR40%X als Zeiger auf eine ISB-Seite so, daß MR eine gerade Anzahl Einsen enthält
- jedes Byte in der oberen Hälfte sollte eine gerade Anzahl Einsen enthalten, -8000-
- jedes Byte in der unteren Hälfte sollte eine gerade Anzahl Einsen enthalten, -007D-

Setzen von DCR <29> (DMP-Bit). Löschen von UACR <4> (SKRBUS-to-ISB-Error-Interrupt-Enable).
Ausführen eines Datenlesevorganges vom ISB zum ISB über den SKRBUS unter Verwendung von MR40. DDP wird verwendet.
Verkleinern von BR auf 3. Es dürfte kein Interrupt erfolgen. Jetzt wird UACR <4> gesetzt - es müßte ein Interrupt auf der Adapterstufe erfolgen.
Interrupt-Service-Routine:
Lesen von BRRVR(N), N entspricht der Adapterstufe. Bit 31 sollte gesetzt sein; <30:00> sollte gelöscht sein. Lesen der BRRVR-Full-Bits - sollten gelöscht sein. Löschen DCR, UACR und Rücksprung.
Lesen DPR1 - sollte gelöscht sein. Lesen der empfangenen Daten - sollten Null sein. Lesen UASR <3> (MRPF) - sollte gesetzt sein. Lesen UASR <0> (SSYNTO) - sollte gesetzt sein.

Subtest 2 --- Laden von MR40%X mit einer ungeraden Anzahl Einsen. Setzen DCR <29>, UACR <4>. Nochmaliges Ausführen des gleichen Ladevorganges - es dürfte kein Interrupt auftreten.

14.6.43. Test 43 - ISB-Interrupt-Test

Dieser Test prüft, ob bei der Ausführung eines Lesevorganges im W/A-Modus unter Verwendung eines Map-Registers, das eine gerade Anzahl von Einsen in irgendeinem Byte der oberen und unteren Hälfte enthält, wobei das Defeat-Map-Parity-Bit <29> in DCR gesetzt wird, ein Interrupt erfolgt.
Es werden alle Kombinationen von:
-- ein Byte in der oberen Hälfte und
-- ein Byte in der unteren Hälfte von MR gerade, getestet.

Hierbei wird BDP verwendet.

Voraussetzungen:
MR 40%X, BDP und UACR sind in Ordnung.
Subtest 1, Error 1 und 2 - MR u g u u (u=ungerade, g=gerade)
" 2, " 1 " 2 - MR g u u u
" 3, " 1 " 2 - MR u u g u
" 4, " 1 " 2 - MR u u u g
" 5, - MR u u u u - es dürfte kein Interrupt erfolgen.

Testschritte:
Verwenden der BDP.

Subtest 1 --- MR40=Zeiger auf eine ISB-Seite. Setzen von DCR <29>, Löschen UACR <4>. Daten werden vom ISB über den SKRBUS unter Verwendung dieses MR40 zum ISB gelesen - BDP1 wird verwendet.Verkleinern von BR auf 3. Es dürfte kein Interrupt erfolgen.
Jetzt wird UACR <4> gesetzt - es müßte ein Interrupt in der Adapterstufe auftreten.
Interrupt-Service-Routine:
Lesen von BRRVR(N), wobei N der Adapterstufe entspricht. Bit 31 sollte gesetzt sein; <30:00> sollte gelöscht sein.
Lesen der BRRVR-Full-Bits -sollten gelöscht sein.
Löschen DCR, UACR - Rücksprung.
Lesen DPR1 - sollte gelöscht sein.
Lesen der empfangenen Daten - sollten Null sein.
Lesen UASR <3> - sollte gesetzt sein.
Lesen UASR <0> - sollte gesetzt sein.

Subtest 2 --- Verändern des Map-Registers und Wiederholen des gesamten Ablaufes - es müßte ein Interrupt auftreten.

Subtest 3 --- Verändern von MR. Lesen vom ISB-Speicher über den SKRBUS und MR40 - es sollte sich ein Interrupt ergeben.

Subtest 4 --- MR s.o.,Erzeugen eines Buffer-Transfer-Error.
Ausführen eines Schreibvorganges - es dürfte kein Interrupt auftreten. Purge (siehe 14.5.14).
Prüfen, ob DPR1<30> (PTE) gesetzt ist.
Jetzt wird MR so geändert, daß die untere Hälfte von MR eine gerade Anzahl Einsen enthält (7C). Schreiben von 4 Wörtern auf diesen Speicherplatz im W/A-Modus über den gleichen BDP1.
Prüfen, daß kein Datenwert nach FC00 gelangt ist.

Subtest 5 --- Laden von MR40 wie oben beschrieben. Setzen von DCR<29> und UACR<4>. Nochmaliges Lesen - es sollte kein Interrupt erfolgen.

14.6.44. Test 44 - ISB-Interrupt-Test

Es findet eine Prüfung des Purge-Error-Path statt, wobei das Defeat-Map-Parity in dem DCR-Register gesetzt ist und nur die untere Hälfte des Map-Registers gerade ist. Zuerst wird ein Schreiben ausgeführt, dann ein Purge (siehe 14.5.14) wonach, ein Interrupt erfolgen müßte. BDP1 wird verwendet. Außerdem wird getestet, daß bei ungerader unterer und oberer Hälfte von MR und gesetztem Defeat-Map-Parity-Bit nach einem Schreiben und Purge kein Interrupt erfolgt.

Voraussetzungen:
MR 40%X, BDP und UACR sind in Ordnung.

Testschritte:
Prüfung einer weiteren Verbindung zu den Steckeinheiten USI, UCB und UMD.

Subtest 1 --- MR40 als Zeiger auf eine ISB-Seite einstellen so, daß MR eine gerade Anzahl von Einsen enthält.
Prüfen, ob DCR<29> (DMP) gelöscht ist. Lesen von Daten vom ISB zum ISB über den SKRBUS unter Verwendung dieses MR40%X - verwendet wird BDP1. Setzen DCR<29>, dann Purge. Setzen UACR<4>, danach Verkleinern von BR auf BR3 - es sollte ein Interrupt in der Adapterstufe erfolgen.
Interrup-Service-Routine :
Lesen von BRRVR(N), wobei N der Adapterstufe entspricht. Bit 31 sollte gesetzt sein; <30:00> sollten gelöscht sein. Löschen DCR, UACR und Rücksprung.
Lesen DPR1<30> (BTE) - sollte gesetzt sein. Lesen der empfangenen Daten - sollten Null sein. Lesen UASR<3> (MPF) - sollte gesetzt sein. Lesen UASR<0> (SSYNTO) - sollte gesetzt sein.

Subtest 2 --- Laden von MR40 mit einer ungeraden Anzahl von Einsen. Ausführen eines Datenlesens wie zuvor. Setzen von DCR<29> (DMP). Purge. Lesen UASR<3> (Map-Parity-Failure) - sollte gelöscht sein.

14.6.45. Test 45 - ISB-Interrupt-Test

Die Ausführung eines Lesevorganges (Byte-Operation) im W/A-Modus führt zu einem Interrupt, wenn der zu lesende Datenwert gerade ist (1 Byte) und das Defeat-Data-Path-Parity-Bit <28> in DCR gesetzt ist. Das Data-Path-Parity-Error-Bit in UASR wird gesetzt.
Wenn sich der Datenwert auf ungerade in allen Bytes ändert, so erfolgt kein Interrupt und kein Fehler.

Voraussetzungen:
Normales W/A-Lesen, DCR, UACR, DPR und UASR sind in Ordnung.

Testschritte:
1). Setzen von MR40%X als Zeiger auf TXBUF in ISB, verwendet BDP1. Laden eines Langwortes (Daten), daß eine gerade Anzahl von Einsen im untersten signifikanten Byte enthält. TXBUF= 010101AA.
Setzen DCR<28> (Defeat-Data-Path-Parity). Leseoperation (Byte) von TXBUF nach RXBUF über die SKRBUS-Adresse entsprechend MR40. Setzen UACR<4>; Verkleinern von BR auf 3 - es müßte ein Interrupt erfolgen.
Interrup-Service-Routine:
Lesen BRRVR(N) - N entspricht der Adapterstufe. Bit 31 - sollte gesetzt sein; <15:00> sollten gelöscht sein. Rücksprung.
Lesen UASR<5> - sollte gesetzt sein. Lesen DPR(N) das BTE-Bit sollte gesetzt sein. Löschen UACR, DCR, DPR<30> und UASR<5>. Jetzt wird der Datenwert verändert - 0101AA01 und obiger Ablauf wiederholt. Das wird 4mal ausgeführt, wobei der Datenwert nur in einem Byte zur gleichen Zeit gerade ist. Der gesamte Test wird unter Verwendung von BDP2:BDP15 wiederholt.
2). Laden eines Datenwortes mit ungerader Anzahl von Einsen nach TXBUF. Setzen von DCR<28>. Lesen von TXBUF nach RXBUF über den SKRBUS. Lesen von UASR<5> - sollte gelöscht sein.

14.6.46. Test 46 - FMER-Register-Test nach Fehlern

Dieser Test prüft, ob nach einem von 5 Fehlern, den das FMER (Failed-Map-Entry-Register) auffängt, die richtige Map-Register-Nummer, die verwendet wurde, enthalten ist. Dann wird der gleiche Fehler nochmal erzeugt, aber dieses Mal unter Verwendung eines anderen Map-Registers. Es wird getestet, ob FMER noch die alte Map-Register-Nummer enthält.

Testschritte:
Subtest 1 --- Test Selbstzustandsfixierung (Latching) von FMER infolge von IVMR - Ausführen eines W/A-Datenlesens mit MR1 , das nicht validiert ist.
Prüfen, ob IVMR gesetzt ist. Lesen FMER - sollte 1 enthalten.IVMR nicht löschen. Ein anderes W/A-Lesen mit MR40%X ausführen. Lesen von FMER - sollte noch die 1 enthalten. Prüfen LEB in UASR - sollte gesetzt sein. Löschen von UASR und FMER.

Subtest 2 --- Test Selbstzustandsfixierung von FMER infolge CXTO. Setzen MR40 als Zeiger auf einen nicht existierenden ISB-Speicherplatz - MR40 validiert.
Ausführen eines W/A-Lesens mit diesem MR40.
Prüfen, ob UASR <6> (CXTO) gesetzt ist. Prüfen, ob FMER =40%X.
Nochmaliges Lesen von einem nicht existierenden ISB-Speicherplatz mit MR1.
Prüfen, ob FMER noch 40%X ist. Prüfen LEB - sollte gesetzt sein. Warten bis MRPF-Bit in UASR gesetzt worden ist. Prüfen, ob FMER =40%X.
Nochmaliges W/A-Lesen mit MR1. MR1 wie MR40 einstellen - 20us warten Prüfen, ob LEB gesetzt worden ist. Prüfen, ob FMER noch 40%X ist.

Subtest 3 --- Prüfen der Zustandsfixierung von FMER infolge CXTER (Command-Transmit-Error).
Initialisierung des SBA. Schreiben nur eines Wortes nach DCR über den SKRBUS, dann Purge (siehe 14.5.14).
Prüfen, ob UASR<7> (CXTER) gesetzt ist. Ebenso wird geprüft, ob FMER =40%X.
Setzen MR1 als Zeiger auf DCR. Wiederholen des Schreibens, dann Purge.
Prüfen, ob LEB in UASR gesetzt ist und ebenso, ob FMER unverändert 40%X ist.

Subtest 4 --- Test der Zustandsfixierung von FMER infolge MRPF. Initialisierung des SBA.
Setzen von MR40%X als Zeiger auf eine ISB-Seite so, daß ein Byte in der oberen Hälfte eine gerade Anzahl von Einsen (8060) und ein Byte in der unteren Hälfte eine gerade Anzahl von Einsen (007E) enthalten. Setzen von DCR<29>.
Ausführen eines W/A-Lesens unter Verwendung von MR40.

Subtest 5 --- Test der Zustandsfixierung von FMER infolge von DDPP (Defeat-Data-Path-Parity). Initialisierung des SBA.
Laden von AAAAAAAA nach TXBUF - gerade Anzahl von Einsen in allen Bytes. MR40%X als Zeiger auf TXBUF einstelllen. DCR<28> setzen. Zweimaliges Ausführen eines W/A-Lesens von TXBUF nach RXBUF unter Verwendung von MR40%X.
Warten bis UASR<5> gesetzt worden ist.
Überprüfen, ob FMER =40%X.
Nochmaliges Lesen von 2 Wörtern unter Verwendung von MR1.
Prüfen, ob LEB gesetzt worden ist und FMER noch 40%X ist.

14.6.47. Test 47 - Test SET DATI-Funktion

Der Test überprüft ob ein SET existiert. In diesem Fall wird der Test fortgeführt, im anderen Fall beendet (gilt auch für die nachfolgenden Tests; s.a. 14.6.56)

Voraussetzungen:
MR 40%X und BDP1 sind in Ordnung.
Das ist das erste Mal, wo eine Datenübertragung, anstatt im W/A-Modus, zu einem Gerät ausgeführt wird.

Testschritte:
Test, ob eine SET vorhanden ist.
Prüfen der SKRBUS-Integrität der Steckeinheiten UAI und UCB.
Setzen von MR40 als Zeiger auf TXBUF, Verwenden von BDP1.
Löschen von BECR1 <8,9> für die DATI-Funktion.
Laden von TXBUF mit AAAA. Löschen des BEDB-Registers. Laden von BEBA mit der SKRBUS-Adresse entsprechend MR40 und TXBUF. Laden des BECC-Registers mit FFFF für eine Zykluszählung. Setzen des GO-Bit in BECR1.
Prüfen, ob BEDB AAAA enthält.
Lesen von BECR1<15>, BECR2<5,6,8,9,10> (Error-Bits) - sollten gelöscht sein.
Laden von TXBUF mit 256 Datenwörtern. Setzen des BECC-Registers auf 256 (eine Seite) für die Zykluszählung. Setzen von BECR1<0> (GO).
Prüfen, ob BEDB das 256. Wort von TXBUF enthält.

14.6.48. Test 48 - Test SET DAT-I/P-Funktion

Für den SET wird die DAT-I/P-Funktion eingestellt und geprüft, ob BEDB exakt beschrieben worden ist. Es werden 256 Datenübertragungen in BEDB geprüft.

Voraussetzungen:
Der Test wird nur bei vorhandener SET ausgeführt.
MR 40%X, DDP und DATI arbeiten.

Testschritte:
Prüfung der SKRBUS-Integrität für die Steckeinheiten UAI und UCB.
Setzen von MR40%X als Zeiger auf TXBUF, Verwendung von DP0. Setzen von BECR1<8>, Löschen von BECR1<9> für die DAT-I/P-Funktion. Setzen von BECR1<14>. Löschen des BEDB-Registers. Laden des BEBA-Registers mit einer SKRBUS-Adresse entsprechend MR40%X und TXBUF. Laden von BECC mit FFFF für eine Zykluszählung. Setzen des GO-Bit in BECR1.
Prüfen, ob die ursprünglichen Daten in TXBUF infolge der SET-DATO-Funktion nach DAT-I/P um 1 Bit nach links verschoben sind.
Laden des BECC-Registers mit 256 für eine Zykluszählung. Nochmalige Ausführung einer DAT-I/P-Funktion unter Verwendung von DP0.
BEDB sollte das 256. Wort von TXBUF - um ein Bit nach links verschoben - enthalten.

14.6.49. Test 49 - Test SET DATO-Funktion

Dieser Test veranlaßt für die SET die DATO-Funktion und prüft, ob Daten von BEDB nach ISB (RXBUF) gelesen werden.

Voraussetzungen:
Dieser Test wird nur bei vorhandener SET ausgeführt.
MR40%X, BDP1, Purge (siehe 14.5.14), W/A R/W in Ordnung.

Testschritte:
Prüfung der SKRBUS-Integrität der Steckeinheiten UAI und UCB.
Setzen von MR40%X als Zeiger auf RXBUF. Verwenden von BDP1. Setzen von BECR1<8,9> =01 für die DATO-Funktion. Laden des BEDB-Registers mit AAAA. Laden des BEBA-Registers mit der SKRBUS-Adresse entsprechend MR40%X und RXBUF. Löschen von RXBUF Laden von BECC mit FFFF für eine Zykluszählung. Setzen des GO-Bits in BECR1. Purge.
Prüfen, ob RXBUF AAAA enthält.
Prüfen von BECR1<15> und BECR2<5,6,8,9,10> (Error-Bits) - sollten gelöscht sein.

14.6.50. Test 50 - Test SET DATO/B-Funktion

Für den SET wird die DATO/B-Operation eingestellt und geprüft, ob Daten aus BEDB (Byte) nach RXBUF (ISB) geschrieben werden.

Voraussetzungen:
Dieser Test wird nur bei vorhandener SET ausgeführt.
MR40%X, BDP1, Purge (siehe 14.5.14), W/A R/W, DATI und DATO arbeiten.

Testschritte:
Prüfung der SKRBUS-Integrität für die Steckeinheiten UAI und UCB.
MR40 als Zeiger auf RXBUF einstellen, BDP1 verwenden. Setzen von BECR1<8,9>=11 für die DATO/B-Funktion. Laden des BEDB-Registers mit FFAA. Laden des BEBA-Registers mit der SKRBUS-Adresse entsprechend MR40 und RXBUF. Löschen von RXBUF. Laden von BECC mit FFFF für eine Zykluszählung. Setzen des GO-Bits in BECR1. Purge.
Prüfen, ob RXBUF AA enthält.
Prüfen von BECR1<15>, BECR2<5,6,8,9,10> (Error-Bits) - sollten gelöscht sein.

14.6.51. Test 51 - ISB-Interrupt durch den SET

Unterbrechungstest des ISB auf allen Stufen durch den SET.
Die Steckeinheiten UAI und UCB werden getestet.

Voraussetzungen:
Dieser Test wird nur bei vorhandener SET ausgeführt.
SET DATI, UACR, BECR1 sind in Ordnung.

Testschritte:
Setzen von MR40%X als Zeiger auf TXBUF, Verwenden von BDP1. Laden des BEBA-Registers mit der SKRBUS-Adresse entsprechend MR40 und RXBUF. Löschen des BEDB-Registers, Laden von TXBUF mit AAAA. Setzen von BECR1<8,9>=00 für die DATI-Funktion. Setzen von BECR1 für BR7-Interrupt, Interrupt-Enable. Prüfen, ob UACR<5,6> gelöscht sind. Löschen von BRRVR's (alle 4) - sollten Null sein. Laden des Vektors mit der Service-Routinen-Adresse. In BECR1 wird das GO-Bit gesetzt - es dürfte kein Interrupt erfolgen.
Verkleinern von BR auf BR3 - es sollte noch kein Interrupt auftreten.
Setzen UACR<6> (IFS) - noch kein Interrupt.
Setzen UACR<5> (Bus-Request-Interrupt-Enable) - jetzt müßte in BR7 ein Interrupt erfolgen.
Interrup-Service-Routine:
Lesen der BRRVR Full-Bits (alle 4) - sollten Null sein. Lesen des BRRVR7-Registers - sollte den SET-Vektor enthalten. Lesen aller anderen BRRVR - sollten Null sein. Lesen der BRRVR-Full-Bits - sollten Null sein. Löschen UACR<5,6>. Rücksprung.

Überprüfen, daß in jeder anderen Stufe keine Unterbrechung erfolgt.
Nochmaliges Laden von BECC für einen Worttransfer. Setzen UACR<5>. Laden von BECR1 für eine DATI-Funktion, Stufe BR4, Interrupt-Enable und GO-Bit. Verkleinern von BR auf BR3. Es dürfte kein Interrupt auftreten. Setzen von UACR<6> (IFS) - jetzt sollte eine Unterbrechung erfolgen.

Der gesamte Ablauf wird für die Interrupts der Stufen BR6, BR5 und BR4 wiederholt.

14.6.52. Test 52 - Zugriff während der Ausschaltfolge

Auf einen SKRBUS-Netzausfall kann noch auf den SKRBUS zugegriffen werden zwischen ACLO und Init (5ms). Außerdem prüft der Test, ob ein anstehender SKRBUS-Interrupt zwischen ACLO und Init nicht behandelt wird.

Voraussetzungen:
Der Test wird nur bei vorhandener SET ausgeführt.

Testschritte:
Subtest 1 --- Initialisierung des SBA. Setzen des UPF-Bits in UACR. Schreiben eines Wortes in das BEDB-Register im SET. 20us warten. Nochmaliges Schreiben in das BEDB-Register (18 Mal) - bis 5ms nach dem Setzen des UPF-Bits sollte auf den SET zugegriffen werden können. Dann Warten bis das UBINIT-Bit in CNFGR gesetzt worden ist. Nochmaliges Schreiben eines Wortes auf BEDB. Müßte eine Maschinenprüfung (siehe 14.5.14) veranlassen.

Subtest 2 --- Initialisierung des SBA. Setzen von SET so, daß sie den ISB auf BR7 unterbricht. Setzen der Interrupt-Enable-Bits, IFS-Bits (Interrupt-Field-Switch) und des UPF-Bits (SKRBUS-Power-Fail) in UACR. Setzen des GO-Bit im BECR1-Register. Verkleinern von BR auf BR6.
Interrup-Service-Routine:
Lesen von BRRVR7. Die Bits <15:00> dürften nicht gleich dem SET-Vektor sein. Löschen des UPF-Bits.

14.6.53. Test 53 - Disable-Interrupt-Bit und Ablaufblockierung

Die Steckeinheiten UAI, UCB und USI werden dabei geprüft.

Voraussetzungen:
SET muß existieren, sonst wird der Test nicht ausgeführt.

Testschritte:
Initialisierung des SBA. Setzen der SET- und UACR-Register so, daß die SET in BR4 eine Unterbrechung hervorruft. Setzen von DCR<30> (Disable-Interrupt). Verkleinern von BR auf BR3.
Interrup-Service-Routine:
Lesen des BRRVR4-Registers - sollte einen Gerätevektor enthalten.
Rücksprung.
Die SET setzt BR auf BR7. Lesen von DCR<27> - sollte gesetzt sein, um nachzuweisen, daß der SBA nicht ablaufblockiert ist.
Löschen von UACR, DCR<30>.

14.6.54. Test 54 - Contention-Logik-Test

Dieser Test prüft jede mögliche Laufbedingung, wenn alle 4 Auswahl-Leitungen UB_ATT_SEL, SB_SEL, DMA_SEL, FILE_WRITE_SEL - zu etwa der gleichen Zeit eingeschaltet werden.

Voraussetzungen:
Dieser Test wird nur ausgeführt, wenn SET existiert.
Prüfung der Steckeinheit UCB.

Testschritte:
Es wird versucht die 4 verschiedenen Auswahlbedingungen zu etwa der gleichen Zeit einzuschalten und geprüft, ob die Contention-Logik arbeitet.

A) Laden von MR40%X mit einer TXBUF-Seite, validieren des MR, Auswahl von BDP1 - für Teil C.
Setzen von UACR<4,5,6> aller Interrup-Enable-Bits.

B) SET so einstellen, daß sie kontinuierlich arbeitet, wobei UB_SEL und DMA_SEL eingeschaltet werden. Laden des BECC-Registers für die Zählung von 100 Wörtern. Setzen von BECR1 - BR6, DATI, Interrupt-Enable. Löschen des BEDB-Registers. Laden des BEBA-Registers mit einer SKRBUS-Adresse (1000%X), von dort sollen Daten gelesen werden. MR20%X auf eine SKRBUS-Adresse (4000%X) einstellen, so daß die Daten effektiv vom ISB-Speicher nach TMPBUF gelesen werden können. Laden des Vektors für die Stufe BR7 mit der Fehler-Service-Routinen Adresse ISR_ERR.
In dieser Routine wird das BRRVR7-Register gelesen, ein Error-Flag gesetzt und zurückgesprungen.
Jetzt wird der Vektor für BR6 mit einer anderen Service-Routinen-Adresse geladen ISR_CON.
In dieser Routine wird BRRVR6 gelesen - das Register sollte den SET-Vektor enthalten. Setzen eines Flag, das das Auftreten eines Interrupts anzeigt.
Setzen BECR1 - BR6, DATI, Interrupt-Enable, GO. Rücksprung.
Ausführung eines SET-Schreibens vom ISB durch Setzen des GO-Bits in BECR1.
Verkleinern von BR auf BR5 - es sollte ein Interrupt erfolgen. Nach dem Rücksprung wird geprüft, ob das Interrup-Flag gesetzt war.
Das Flag wird gelöscht. Ebenso wird BRRVR6 auf das Vorhandensein des SET-Vektors getestet, indem das Error Flag 2 geprüft wird.

C) Einschalten von "FILE_WRITE_SEL" und "SB_SELECT". Lesen eines Wortes von TXBUF nach RXBUF über den SKRBUS unter Verwendung von MR40%X. Lesen des UASR-Registers (SB_SELECT). Vergleich der Daten in RXBUF mit TXBUF.

Wiederholen des obigen Ablaufs (ab C) 10000 mal.
Dann wird durch Löschen des GO-Bits und des Interrupt-Enable in BECR1 angehalten.
Prüfen, ob BEDB das 100. Wort von TMPBUF enthält.

14.6.55. Abkürzungsverzeichnis

DDP	direkter Datenweg
BDP	gepufferter Datenweg
W/A	Wrap around
MR	Map-Register
SBA	SKRBUS-Adapter
SET	SKRBUS-Exerciser
IB Bus	interner Bus

ISB	Interner Synchron Bus
USI	Steckeinheit des SBA
UAI	" " "
UCB	" " "
UMD	" " "
RXBUF	Receive Buffer
TXBUF	Transmitting Buffer
%D	dezimal
%X	hexadezimal
R/W	read/write
< >	kennzeichnet Bitnummer
( )	" Bitbezeichnung
BR	Prozessorprioritätsstufen
S/B	SHOULD BE
CTL	CONTROL
ISR4	Interrupt-Service-Routine 4

14.6.56. Auflistung der Tests

Test 1: READ/WRITE CNFGR REGISTER
Test 2: READ/WRITE DCR REGISTER
Test 3: TEST DCR and CNFGR DUAL ADDRESS ABILITY
Test 4: READ/WRITE UACR REGISTER
Test 5: READ/WRITE UASR REGISTER
Test 6: INITIALIZATION TEST BY SETTING ADINIT
Test 7: SKRBUS POWER FAIL SEQUENCE TEST
Test 8: TEST THAT ADINIT TIMING and POWER FAIL SEQUENCE OK
Test 9: TEST CONFIGURATION INTERRUPT LOGIC
Test 10: TEST SKRBUS MASK ERROR FUNCTION
Test 11: TEST SBA MASK ERROR FUNCTION
Test 12: READ/WRITE ALL MAPS
Test 13: ADDRESS TEST OF ALL MAP REGISTERS
Test 14: READ/WRITE ALL BRSVR'S and ADDRESS TEST OF BRSVR'S
Test 15: TEST SSYNTO-BIT IN UASR
Test 16: TEST FUBAR REGISTER
Test 17: TEST FMER REGISTER and IVMR BIT IN UASR
Test 18: TEST DPR REGISTER ACCESSABILITY
Test 19: TEST MAP REGISTER DISABLE FUNCTION
Test 20: TEST OF STATE BITS IN DPR'S WITHOUT ISB WRITE
Test 21: DATA TRANSFER TEST - WRITE
Test 22: TEST PURGE DOING A NOP
Test 23: TEST OF DPR MASK BITS and DATA XFER - 8 BYTES
Test 24: NO SHORTS OR OPENS IN ADDRESS LINES TO (MASK) RAMS
Test 25: BDP ADDRESS TEST
Test 26: TEST THAT BUSY CONFIRMATION REALLY WORKS
Test 27: CHECK SKRBUS WITH SEVERAL DIFFERENT DATA PATTERNS
Test 28: ALIGNMENT TEST DURING WRITE
Test 29: ALIGNMENT TEST DURING WRITE - WITH BYTE OFFSET SET
Test 30: DATA TRANSFER TEST - READ
Test 31: ALIGNMENT TEST DURING READ
Test 32: MAP BOUNDARY CROSSING DURING WRITE
Test 33: MAP BOUNDARY CROSSING DURING READ
Test 34: TEST I/O DATO FUNCTION
Test 35: TEST I/O DATO/B FUNCTION
Test 36: TEST I/O DATI (B01)
Test 37: TEST I/O DATI (B23)
Test 38: TEST INTERLOCK LOGIC - DAT I/P FUNCTION

Test 39: TEST COMMAND ADDRESS ERROR
Test 40: TEST COMMAND ADDRESS T/O ERROR
Test 41: TEST LOST ERROR BIT FUNCTION
Test 42: TEST SBA INTERRUPTING ISB USING DEF. MAP. PAR. FUNCTION
Test 43: SBA INTERRUPTING ISB USING DEF. MAP. PAR. FUNCTION DMA ERROR PATH
Test 44: SBA INTERRUPTING ISB USING DEF. MAP PARITY FUNCTION PURGE ERROR PATH
Test 45: SBA INTERRUPTING ISB USING DEF. DAT. PATH. PARITY
Test 46: TEST FMER LATCHES BY ERROR SIGNALS
Test 47: TEST SET DATI FUNCTION
Test 48: TEST SET DAT I/P FUNCTION
Test 49: TEST SET DATO FUNCTION
Test 50: TEST SET DAT O/B FUNCTION
Test 51: SET INTERRUPTING ISB AT ALL LEVELS
Test 52: ACCESS OF SKRBUS DURING POWER DOWN SEQUENCE
Test 53: DISABLE INTERRUPT BIT IN DER and SBA DOESN'T HANG UP
Test 54: CONTENTION LOGIC TEST

Bei nicht vorhandener SET gilt für die Testausschriften ab Test 47:

Test 47: TEST SET DATI FUNCTION - NOT EXECUTED
Test 48: TEST SET DAT I/P FUNCTION - NOT EXECUTED
Test 49: TEST SET DATO FUNCTION - NOT EXECUTED
Test 50: TEST SET DAT O/B FUNCTION - NOT EXECUTED
Test 51: SET INTERRUPTING ISB AT ALL LEVELS - NOT EXECUTED
Test 52: ACCESS OF SKRBUS DURING POWER DOWN SEQUENCE - NOT EXECUTED
Test 53: DISABLE INTERRUPT BIT IN DER & SBA DOESN'T HANG UP - NOT EXECUTED
Test 54: CONTENTION LOGIC TEST - NOT EXECUTED

15. Bus-Interaktion-Test

Erstellung erfolgt zu einem späteren Zeitpunkt.

16. K 1840 System-Test

Erstellung erfolgt zu einem späteren Zeitpunkt.

17. Diagnose-Supervisor

17.1. Einführung

Der Diagnose-Supervisor (kurz Supervisor) (Dok.-Nr. 1.57.700000.1) erlaubt, Diagnoseprogramme der Ebenen 2, 2R und 3 abzuarbeiten. Programme der Ebene 2R werden im User-Modus (Supervisor unter Steuerung des SVP), Programme der Ebene 3 im Stand-alone-Modus abgearbeitet. Programme der Ebene 2 arbeiten in beiden Modi.
Der Supervisor stellt einen Satz von Kommandos und Steuerflags zur Verfügung, die es dem Operator gestatten, die Diagnoseprogramme zu laden und abzuarbeiten.
Für eine tiefgründige Testung des Rechnersystems ist die genaue Kenntnis der Funktion der Kommandos und Steuerflags Voraussetzung.

Die Supervisor-Kommandos können in 4 Gruppen eingeteilt werden.

Programm- und Testablaufsteuerung
Kommandos für Scripting
Abarbeitungssteuerung
Debug- und Dienstprogrammkommandos

Kommandos, Schalter und Literale können soweit abgekürzt werden, solange die Abkürzungen noch eindeutig sind (z.B. LOAD-Kommando: L, START-Kommando: ST).
Der Supervisor meldet sich mit der Eingabeanforderung DS>
Alle Kommandos werden mit <CR> abgeschlossen.
Ein Kommando kann bei Überschreitung der Zeilenlänge ohne Eingabe eines Hinweiszeichens (z.B. Minus) sofort auf der nächsten Zeile fortgesetzt werden.
Als Trennzeichen zwischen den einzelnen Ausdrücken des Kommandos sind ein oder mehrere Leerzeichen zu verwenden.

In der Kommandobeschreibung haben folgende spezielle Zeichen folgende Bedeutung:

< >	schließen symbolische Argumente (numerische Ausdrücke oder Zeichenketten) ein
[ ]	schließen optionale Argumente ein
	ODER-Funktion innerhalb des Kommandos
Literale	(z.B. ALL, OFF und FLAGS) werden großgeschrieben
-	Minuszeichen am Ende einer Zeile dient als Kennzeichen, daß das Kommando auf der nächsten Zeile fortgesetzt wird.
!	Leitet Kommentar ein
<file-spec>	Die Datei-Spezifikation hat komplett folgenden Aufbau: <device>:[<directory>]<file-name>.<typ>;<version> Die Directory-Bezeichnung muß in eckige Klammern eingeschlossen werden. Es können auch Subdirectories in der Form [<directory>.<subdirectory>...] angegeben werden.

17.2. Diagnose- und Supervisor-Kommandos

17.2.1. Programm- und Testablaufsteuerkommandos

Diese Kommandos erlauben es dem Operator, Programme und Programmabschnitte auszuwählen und den Testablauf zu steuern.

SET LOAD-Kommando

SET LOAD [<device>:][<directory>]

Das SET LOAD-Kommando legt das Speichergerät und/oder die Directory fest, von dem der Supervisor die Diagnoseprogramme lädt und in der die Help-Dateien stehen. Als Standard-Ladegerät wird das Gerät eingetragen, von dem der Supervisor gebootet wurde. Im User-Modus (unter SVP) wird das Standard-Ladegerät und die Directory eingetragen, das äquivalent dem Namen SYS$SYSROOT plus dem Ausdruck [SYSMAINT] ist. Im Normalfall ist dies die Systemplatte plus [SYS0.SYSMAINT], d.h. z.B. DBA0:[SYS0.SYSMAINT].
Dieses Kommando wird somit nur bei Änderung des Speichergerätes und im Zusammenhang mit dem LOAD- oder dem RUN-Kommando verwendet.

Beispiele:

```
DS> SET LOAD DBA0:[SYS0.SYSMAINT]
DS> LOAD PVTAA

DS> SET LOAD DBA0:[SYS0.SYSMAINT]
DS> RUN PVTAA
```

SHOW LOAD-Kommando

SHOW LOAD

Der Supervisor zeigt den Namen des Speichergerätes und der Directory an, von denen die Diagnoseprogramme mit Hilfe des LOAD-Kommandos geladen werden.

Beispiel:

```
DS> SHOW LOAD
DBA0:[SYS0.SYSMAINT]
DS>
```

DIRECTORY-Kommando

DIRECTORY [<file-spec>] [<qualifiers>]

Dieses Kommando zeigt den Inhalt der spezifizierten Directory an. Wurde diese nicht spezifiziert, wird die Standard-Directory (siehe SET LOAD-Kommando) ausgelistet. Der Datei-Name und/oder -typ kann durch die Wildcard-Zeichen "*" oder "%" ersetzt werden. Werden diese nicht spezifiziert, wird automatisch " " eingesetzt. Das Gerät muß vorher angeschlossen worden sein (siehe ATTACH).

Beispiele:

```
DS> DIRECTORY DBA0:[SYS0.SYSMAINT]
        .
        .
        .
DS> DIRECTORY DBA0:
        .
        .
        .
DS> DIRECTORY
        .
        .
        .
DS>
```

/WIDE

Der Inhalt wird untereinander ausgegeben. Dies ist erforderlich, wenn die Dateinamen aus vielen Zeichen (max. 252 einschließlich Sonderzeichen) bestehen.

LOAD-Kommando

LOAD <file-spec>

Dieses Kommando lädt die spezifizierte Datei (d.h. ein Diagnoseprogramm) in den Hauptspeicher. Wurden Gerät und Directory nicht angegeben, werden die Standardwerte bzw. die genommen, die durch das SET LOAD-Kommando festgelegt wurden. Wird der Dateityp nicht angegeben, wird der Standardtyp .EXE verwendet. Wird die Version nicht spezifiziert, wird die Datei mit der höchsten Versionsnummer geladen.
Es steht immer nur ein Diagnoseprogramm im Speicher. Mit dem Laden eines Programmes wird das vorher geladene Programm überschrieben.

Beispiel:

```
DS> LOAD PVTAA              !Laden des Terminal-Diagnoseprogramms
DS>
```

SHOW TESTS-Kommando

SHOW TESTS

Es werden die Nummern und die Namen der einzelnen Tests des geladenen Diagnoseprogramms ausgegeben.

SHOW SECTIONS-Kommando

SHOW SECTIONS

Es werden die Namen der Sektionen des geladenen Diagnoseprogramms angezeigt. Ist kein Programm geladen, erfolgt eine Fehlerausschrift.

ATTACH-Kommando

ATTACH <UUT-type> <link-name> <generic-device-name>

Bevor ein Diagnoseprogramm gestartet wird, muß der Operator mit Hilfe verschiedener ATTACH-Kommandos die zum Test genutzten Hardwareeinheiten (unit under test - UUT) anschließen und damit dem Supervisor bekanntmachen. Jede UUT wird einmalig angeschlossen durch ihre physische Bezeichnung und durch den Anschluß zur übergeordneten Funktionsgruppe. Die übergeordnete Funktionsgruppe muß vorher bereits angeschlossen worden sein (mit Ausnahme von HUB).

1. Parameter <UUT-Typ>:
   Hardware-Bezeichnung des Gerätes bzw. der Funktionsgruppe z.B. RHA40, DWA40, BM16, BE06, PP05, AMF18A, TG911

2. Parameter <link-name>:
   Allgemeiner Gerätename des Gerätes bzw. der Funktionsgruppe, an die die UUT angeschlossen ist.

z.B.	RHA40	an HUB (entspricht ISB)
	DWA40	an HUB
	BM16	an DWn
	BE06	an MEa
	PP05	an RHa
	AMF18A	an DWn
	TG911	an TXB

3. Parameter <generic-device-name>:
   Allgemeiner Gerätename des Gerätes bzw. der Funktionsgruppe, die angeschlossen wird. Der Parameter hat die allgemeine Form ggan: als Gerätename.

gg	zwei Buchstaben des allgemeinen Gerätenamens
a	ein Buchstabe, der den Gerätecontroller spezifiziert
n	eine Ziffer im Bereich 0...255, die ein Gerät aus einer Summe mehrerer gleicher Geräte an der gleichen Link-Baugruppe spezifiziert.

Für einige Hardwarekomponenten sind zusätzlich Informationen erforderlich (z.B. TR-Nummer und BR-Ebene für den RHA40, Kontroller-Nummer für den BM16, CSR-Vektor und BR-Ebene für ein SKRBUS-Gerät). Werden die zusätzlich erforderlichen Informationen nicht eingegeben, erfolgt automatisch eine Eingabeanforderung vom Supervisor.
Die erforderlichen Eingaben sind den jeweiligen Diagnoseprogrammbeschreibungen zu entnehmen. Das ATTACH-Kommando kann mit CTRL/C abgebrochen werden.

Beispiele:

```
DS> ATTACH DWA40 HUB DW0 3 4      !Anschließen des SKRBUS-Adapters
DS> ATTACH AMF18A DW0 TXB         !Anschließen des AMF18-Asynchron-Ports
CSR? 760414                       !Eingabeanforderungen
VECTOR? 340                       !vom Supervisor
BR? 5
ACTIVE LINES? 377
BAUD RATE? 9600
LOOPBACK TYPE? INTERNAL
SKRBUS INIT JUMPER? NO
DS> ATTACH TG911 TXB TXB6         !Anschließen des TG911
DS>
```

DEATTACH-Kommando

DEATTACH <generic-device-name> /ADAPTER=<link-name>

oder

DEATTACH /ADAPTER=<link-name> <generic-device-name>

Das DEATTACH-Kommando bewirkt das Gegenteil des ATTACH-Kommandos, d.h. das spezifizierte Gerät wird abgetrennt. Zu beachten ist, daß beim Abtrennen eines übergeordnete Gerätes alle daran angeschlossenen Geräte mit abgetrennt werden. Es wird eine Mitteilung ausgegeben, welche Geräte abgetrennt wurden.

Beispiele:

```
Angeschlossen wurden:
  DWA40     HUB   DW0
  AMF18A    DW0   TXB
  TG911     TXB   TXB1
  TG911     TXB   TXB2
  TG911     TXB   TXB3

DS> DEATTACH/ADAPTER=TXB TXB3
 TXB3 has been de-attached.
DS> DEATTACH/ADAPTER=HUB DW0
 TXB1 has been de-attached.
 TXB2 has been de-attached.
 TXB  has been de-attached.
 DW0  has been de-attached.
DS>
```

SELECT-Kommando

SELECT [<qualifiers>] <generic-device-name>[:][,<generic-device-name>[:]...] !ALL

Das zu testende Gerät ist nach dem ATTACH-Kommando mit dem SELECT-Kommando und dem allgemeinen Gerätenamen auszuwählen. Werden mehrere ausgewählt, werden sie in der Reihenfolge der Auswahl nacheinander in den Test einbezogen.

/ADAPTER=<name>

Dieser Qualifier schränkt das auszuwählende Gerätespektrum auf die Geräte ein, die an das Gerät, spezifiziert mit "name", angeschlossen sind.

/NOALLOCATE

Ist dieser Qualifier angegeben, wird das ausgewählte Gerät nicht dem Diagnosesystem zugeteilt. Dies wird genutzt, wenn das auszuwählende Gerät bereits einem anderen Nutzer zugeteilt wurde. Nach Freigabe des Gerätes durch den anderen Nutzer kann dieses Gerät sofort in den Test einbezogen werden.

Beispiele:

```
DS> SELECT TXB6
DS> SELECT/ADAPTER=DW0 ALL
DS> SELECT/NOALLOCATE TXB1:
DS>
```

DESELECT-Kommando

```
DESELECT [<qualifiers>] <generic-device-name>[:]-
         [,<generic-device-name>[:]...] !ALL
```

Das spezifizierte Gerät wird aus der Liste der zu testenden Geräte gestrichen. Dazu muß es vorher ausgewählt worden sein.

/ADAPTER=<name>

Diese Qualifier schränkt das zu inaktivierende Gerätespektrum auf die Geräte ein, die an das Gerät, spezifiziert mit "name", angeschlossen sind.

Beispiele:

```
DS> DESELECT TXB6:
DS> DESELECT /ADAPTER=DW0  ALL
DS>
```

SHOW DEVICE-Kommando

SHOW DEVICE [<qualifiers>] <generic-device-name>[:][,<generic-device-name>[:]...]

Dieses Kommando veranlaßt den Supervisor, die charakteristischen Parameter der spezifizierten Geräte auf dem Operatorterminal auszugeben. Sind keine Geräte spezifiziert, werden die Parameter aller angeschlossenen Geräte (mit ATTACH-Kommando) ausgegeben.

/BRIEF

Mit diesem Qualifier wird veranlaßt, daß nur die allgemeinen Gerätenamen und die Hardware-Bezeichnungen ausgegeben werden.

/ADAPTER=<name>

Dieser Qualifier schränkt das auszugebende Spektrum von Geräten auf die ein, die an das Gerät, spezifiziert mit "name", angeschlossen sind.

Beispiel:

```
DS> SHOW DEVICE
_DW0  DWA40  HUB  40F30000
_TXB  AMF18A  _DW0  40FFE10C  CSR=760414(0)
VECTOR=000340(0)  BR=5.  ACTIVE LINES=000040(0)
BAUD RATE=9600  LOOPBACK TYPE=INTERNAL
SKRBUS  INIT  JUMPER=NO
_TXB6  TG911  _TXB  00000000
DS>
```

SHOW SELECTED-Kommando

SHOW SELECTED [<qualifiers>]

Es werden die charakteristischen Parameter der Geräte auf dem Operatorterminal ausgegeben, die mit dem SELECT-Kommando ausgewählt wurden. Das Ausgabeformat entspricht dem des SHOW DEVICE-Kommandos.

/BRIEF

Mit diesem Qualifier wird veranlaßt, daß nur die allgemeinen Gerätenamen und die Hardware-Bezeichnungen ausgegeben werden.

START-Kommando

START [/SECTION:<section-name>][/TEST:<first>[:<last>!/SUBTEST:<num>]]-
[/PASSES:<count>][<qualifiers>]

Das START-Kommando veranlaßt den Supervisor, die Steuerung an die angewählten Tests des Diagnoseprogramms im Hauptspeicher zu übergeben, um die Programmabarbeitung zu beginnen. Dies setzt voraus, daß mit dem LOAD-Kommando ein Diagnoseprogramm vorher geladen wurde.
Jedes Diagnoseprogramm besteht aus diskreten Tests. Diese Tests sind in Sektionen zusammengefaßt in Abhängigkeit von ihrer Funktion, Abarbeitungszeit und ob Bedienerhandlungen erforderlich sind.
Wird im START-Kommando keine Sektion angegeben, werden die Tests der Standard-Sektion abgearbeitet. Tests der Standard-Sektion erfordern keine Bedienerhandlungen. Wird im START-Kommando eine Sektion angegeben, werden die Tests abgearbeitet, die dieser Sektion zugeordnet sind.
Die Eingabe TEST wird in zwei verschiedenen Formen genutzt. Wenn die Argumente "first" und "last" spezifiziert sind, werden nacheinander die Tests, beginnend mit "first" und endend mit einschließlich "last", abgearbeitet. Wird das Argument "first" in Verbindung mit SUBTEST angegeben, wird der Test, spezifiziert mit "first", vom Beginn an gestartet, und es werden soviel Subtests dieses Tests abgearbeitet, wie unter SUBTEST spezifiziert. Wird an diese Kommandoform die

Eingabe PASSES angeschlossen, so wird der letzte Subtest sovielmal durchlaufen, wie durch PASSES spezifiziert wurde.
Wenn kein Test spezifiziert wurde, werden alle Tests der ausgesuchten Sektion (mit /SECTION: spezifiziert oder die Standard-Sektion) abgearbeitet. Ist in TEST nur ein Argument spezifiziert, werden alle Tests, beginnend ab dem spezifizierten bis einschließlich dem mit der höchsten Nummer, der ausgesuchten Sektion abgearbeitet.
In die Abarbeitung werden nur die Tests einbezogen, die Bestandteil der ausgesuchten Sektion sind. Mit der Eingabe PASSES kann die Anzahl der Durchläufe spezifiziert werden. Ohne Spezifizierung erfolgt ein einmaliger Durchlauf (Standardwert ist 1). Der Maximalwert bei PASSES ist 0 und bedeutet unendlich viele Durchläufe. Alle Zahlen sind dezimal einzugeben.

Beispiele:

```
DS> START                              !Abarbeitung des Diagnoseprogrammes im
                                       !Hauptspeicher
DS> START /SEC:MANUAL                  !Abarbeitung der MANUAL-Sektion des Diag-
                                       !noseprogrammes
DS> START /SEC:MANUAL /TEST:32:33      !Abarbeitung der Tests 32 und 33, sofern
                                       !sie in der MANUAL-Sektion enthalten sind
DS> START /TEST:6:12                   !Abarbeitung der Tests 6,7,8,9,10,11,
                                       !12 der Standard-Sektion
DS> START /TEST:9 /SUBTEST:5           !Abarbeitung Test 9, Subtest 1,2,3,4,5
DS> START /TEST:9                      !Abarbeitung Test 9 bis n, wobei n der
                                       !letzte Test der Standard-Sektion ist
DS> START /PASS:3                      !Abarbeitung der Standard-Sektion mit 3
                                       !Durchläufen
DS> START /TEST:9 /SUBTEST:5/PASS:0    !Abarbeitung Test 9, Subtest 1,2,3,4
                                       !und dann Subtest 5 mit unendlich viel
                                       !Durchläufen
```

/QA

Der Supervisor bietet die Möglichkeit, mit Hilfe dieses Qualifier sogenannte Qualitätssicherungsprüfungen auszuführen. Wenn dieser Qualifier gesetzt ist, durchläuft das Diagnoseprogramm verschiedene Schleifen und Tests. Das Programm wird abgearbeitet, bis irgendein QA-Fehler festgestellt wurde oder bis die QA-Prüfung abgearbeitet wurde. Ziel ist eine Automatisierung des Testablaufs.
Die Qualifiers /PASSES, /SECTION und /SUBTEST werden ignoriert. Es werden nur Tests der Standard-Sektion abgearbeitet. Im Einzelnen wird mit /QA folgender Testablauf organisiert:

1. Normaltest
   1.1 Neustart des Diagnoseprogrammes
   1.2 Abarbeitung der gewählten Tests der Standard-Sektion
   1.3 Endemeldung

2. QAMULTIPLEPASS
   2.1 Neustart
   2.2 Abarbeitung der gewählten Tests der Standard-Sektion
   2.3 Endemeldung
   Weiter in der Reihenfolge 2.2, 2.3, 2.2, 2.3, ...
   Diese Schleife wird so oft abgearbeitet, wie durch das Flag QAMULTIPLEPASS vorgegeben wird.

3. QATESTLOOPS
   3.1 Neustart
   3.2 Erster spezifizierter Test wird so oft abgearbeitet, wie durch das QATESTLOOPS-Flag festgelegt wird. Danach wird der zweite Test n-mal abgearbeitet, usw.
   3.3 Endemeldung

4. QASUBTESTLOOPS
   4.1 Neustart
   4.2 Erster Subtest wird so oft abgearbeitet, wie durch das QASUBTESTLOOPS-Flag festgelegt ist.
   4.3 Nachfolgender Subtest wird n-mal abgearbeitet, wobei die vorhergehenden Subtests innerhalb des gleichen Tests wiederholt werden. Beim Abarbeiten des ersten Subtests des nachfolgenden Tests werden alle Subtest des gerade beendeten nochmals abgearbeitet.
   4.4 Endemeldung

5. QUACKLOOPLOOPS
   5.1 Neustart
   5.2 Einmalige Abarbeitung der spezifischen Tests in umgekehrter Reihenfolge
   5.3 Endemeldung

Beispiel:

QAM=3, QAT=4, QAS=3

Diagnoseprogramm:	Test 1 mit drei Subtests Test 2 mit zwei Subtests
Testablauf:	
1, 2	(Normalstart)
1, 2, 1, 2, 1, 2	(QAMULTIPLEPASS)
1, 1, 1, 1, 2, 2, 2, 2	(QATESTLOOPS)
1.1, 1.1, 1.1,	(QASUBTESTLOOPS)
1.1, 1.2, 1.2, 1.2,	
1.1, 1.2, 1.3, 1.3, 1.3,	
1.1, 1.2, 1.3, 2.1, 2.1, 2.1,	
2.1, 2.2, 2.2, 2.2,	
2.1, 2.2	
2, 1	(QACKLOOPLOOPS)

RUN-Kommando

```
RUN <file-spec>[/SECTION:<section-name>][/TEST:<first>[:<last>!/SUBTEST:<num>]]-
              [/PASSES:<count>][<qualifiers>]
```

RUN ist das Äquivalent zur LOAD- und START-Kommandofolge. Die Bedeutung der Eingabe ist mit der des START-Kommandos identisch.

/QA

Dieser Qualifier hat die gleiche Funktion wie beim START- Kommando.

Beispiele:

```
DS> RUN PVTAA          !Laden und Abarbeiten des Diagnoseprogrammes für
                       !lokale Terminals
```

weitere Beispiele analog START-Kommando

CONTROL-C-Kommando

CTRL/C

Normalerweise wird die Steuerung vom Diagnoseprogramm zum Kommandozeilen-Interpreter des Supervisors zurückgegeben. Der Supervisor befindet sich dann in einem Kommandowartezustand und meldet sich mit DS> auf dem Operatorterminal. Der Operator kann jetzt irgendein zugelassenes Kommando eingeben. CTRL/C ist das einzige Supervisorkommando, das man während der Abarbeitung eines Programmes eingeben kann. Arbeitet das Diagnoseprogramm im Konversations-Modus kehrt die Steuerung nach Eingabe CTRL/C zum Kommando-Interpreter innerhalb des Programms für den Konversations-Modus zurück.

SUMMARY-Kommando

SUMMARY

Mit diesem Kommando wird ein Programm des Supervisors gestartet, das den Statistical Report ausgibt. Dieses Kommando wird im allgemeinen nach Abarbeitung des Diagnoseprogramms ausgeführt. Es kann aber sinnvoll sein, den Statistical Report während der Abarbeitung des Diagnoseprogramms ausgeben zu lassen. Dazu ist CTRL/C einzugeben, um in den Kommandozeilen-Interpreter zurückzukehren. Danach ist das SUMMARY-Kommando einzugeben. Mit dem CONTINUE-Kommando kann die Programmabarbeitung fortgesetzt werden.
Der Statistical Report beinhaltet den Namen und die Ausgabe des Diagnoseprogramms, die Summe aller durch das Programm (Hard, Soft, System, Device) und durch den Supervisor festgestellten Fehler.

SHOW STATUS-Kommando

SHOW STATUS

Es werden Name, Ausgabe, Datum, Uhrzeit, Sektion, Fehlerzählzelle, Test, Subtest, Durchlaufzähler und PC-Inhalt des mit CTRL/C unterbrochenen Programms ausgegeben. Wird kein Programm abgearbeitet, erfolgt eine Fehlerausschrift.

CONTINUE-Kommando

CONTINUE

Dieses Kommando veranlaßt die Programmfortsetzung an dem Punkt, an dem es unterbrochen wurde. Das Kommando wird genutzt nach Unterbrechungspunkt, Fehlerhalt oder nach den Kommandos SUMMARY, CTRL/C.

Beispiele:

```
                ...Programm läuft...

   CTRL/C                    !Operator gibt CTRL/C ein.
   DS> SUMMARY               !Eingabeanforderung, Operator fordert
                             !Statistical Report.
                Statistical Report
   DS> CONTINUE              !Eingabeanforderung, Operator weist Fortsetzung
                             !der Programmabarbeitung an
                ...Programm läuft...
```

ABORT-Kommando

ABORT

Dieses Kommando veranlaßt einen Sprung in die Cleanup-Routine mit anschließender Rückkehr zum Supervisor, der in den Kommandowarte-Status eintritt und sich mit DS> meldet. Der Operator kann irgendein Kommando mit Ausnahme CONTINUE eingeben.

Beispiele:

```
                ...Programm läuft...

   CTRL/C                    !Operator gibt CTRL/C ein.
   DS> SUMMARY               !Eingabeanforderung, Operator fordert
                             !Statistical Report.
                Statistical Report
   DS> ABORT                 !Eingabeanforderung, Operator weist Rücksetzen
                             !und Beenden des Programms an.
   DS>
```

EXIT-Kommando

EXIT

Im User-Modus wird die Arbeit im Diagnosesystem beendet, das Betriebssystem meldet sich mit $. Alle ausgewählten Geräte werden automatisch inaktiviert. Im Stand-alone-Modus geht das Programm nach Reinitialisierung des Rechners in Halt. Mit CONTINUE von der Konsole kann in den Kommando-Modus des Supervisors zurückgesprungen werden.

CONTROL-Z-Kommando

CTRL/Z

Dieses Kommando hat die gleiche Wirkung wie das EXIT-Kommando.

CONTROL-Y-Kommando

CTRL/Y

Die Arbeit des Diagnosesystems wird über Interrupt abgebrochen. Im User-Modus ist die Anwendung nicht sinnvoll, da z.B. die Standard-Directory nicht wieder

zurückgestellt wird.

HELP-Kommando

HELP

Der Supervisor gibt einen Überblick über die Möglichkeiten und die Nutzung der Help-Dateien.

HELP <topic>

Der Supervisor gibt entsprechend dem spezifizierten Topic Informationen aus, die er der Help-Datei PVSAA.HLP entnimmt.

HELP <diagnostic name> [<topic> [<subtopic>...]]

Es werden Informationen der Help-Datei des spezifizierten Diagnoseprogrammes und des entsprechenden Topics ausgegeben.

Beispiel:

```
DS> HELP PVTAA


DS>
```

Bemerkung:

Die Help-Dateien müssen in der Standard-Directory oder in der Directory stehen, die mit dem SET LOAD-Kommando festgelegt wurde. Die Eingaben können durch Wildcard-Zeichen ersetzt werden.

MEMORY MANAGEMENT-Kommando

Diese Kommandos sind nicht im User-Modus nutzbar.

SET MM ON
Einschalten Speicherverwaltung

SET MM OFF
Auschalten Speicherverwaltung

SHOW MM
Anzeige, ob Speicherverwaltung ein- oder ausgeschaltet ist

SET MEMORY-Kommando

SET MEMORY <n>

Mit "n" kann die Größe des physischen Speichers (in Seiten) festgelegt werden. Dies kann erforderlich sein, wenn der Supervisor und das Diagnoseprogramm in einem kleineren Speicherbereich arbeiten sollen als dem physisch vorhandenen. Der Wert von "n" muß größer oder gleich Null, aber kleiner als der mögliche physische Speicher sein. Das Kommando arbeitet nicht im User-Modus. Angeschlossene

oder ausgewählte Geräte sind nach Ausführung des Kommandos inaktiviert und freigegeben.

SHOW MEMORY-Kommando

SHOW MEMORY [<qualifiers>]

Mit diesem Kommando kann die Speicherplatzaufteilung des Diagnoseprogrammsystems angezeigt werden.

/MAP

Ausgabe Start- und Endadresse des Diagnoseprogramms, des Supervisors und des dynamischen Speichers sowie die Größe des physischen Speichers. Dieser Qualifier kann auch weggelassen werden.

/DATA_STRUCTURE

Zeigt die internen Datenstrukturadressen des Diagnosesystems einschließlich aller Stacks, Page-Tabellen, SCB und anderer Datenstrukturen (z.B. PCB) an. Die Kenntnis der Datenstruktur kann wertvoll für die Fehlersuche sein. Zu beachten ist, daß sich die Speicheraufteilung im Stand-alone-Modus von der im User-Modus stark unterscheidet, so daß diese im User-Modus nicht angezeigt wird.

/BUFFER

Zeigt den möglichen Platz im P0-, P1- und S0-Adreßraum für Puffer der Diagnoseprogramme sowohl oberhalb, als auch unterhalb des Diagnosesystems an (nur sinnvoll, wenn Diagnoseprogramm geladen ist).
Im User-Modus wird der Speicher oberhalb des Systems nicht angezeigt.

/ALL

Es werden /MAP, /DATA_STRUCTURE und /BUFFER zusammengefaßt.

SET WIDTH-Kommando

SET WIDTH <n>

Mit diesem Kommando kann die Anzahl der Spalten "n" des Operatorterminals eingestellt werden, wenn diese kleiner ist als 132. Ist bei der Ausgabe die Spaltenanzahl erreicht, wird automatisch <CR>, <LF> ausgeführt.

SHOW WIDTH-Kommando

SHOW WIDTH

Es wird die eingestellte Anzahl der Spalten des Operatorterminals angezeigt.

SET PAGE-Kommando

SET PAGE <n>

Mit "n" wird die Größe einer Terminalseite spezifiziert. Sind "n" Zeilen

ausgegeben, wird die Ausgabe gestoppt und es wird erst nach Quittierung einer Eingabeanforderung weiter ausgegeben. Damit wird ein Verlust der ausgegebenen Daten vermieden. Mit CTRL/C kann die weitere Ausgabe abgebrochen werden. Für einen 24-zeiligen Bildschirm ist "n" gleich 22 zu setzen, um eine maximale Zeilenanzahl ohne Verlust ausgeben zu können. Die restlichen zwei Zeilen werden für die Eingabeanforderung benötigt. Der Standardwert ist Null, d.h. es wird keine Seiteneinteilung vorgenommen.

SET ENFORCE-Kommando

SET ENFORCE

Alle Gerätenamen müssen vollständig angegeben werden. Nach dem Start des Supervisors ist dies der Grundzustand.

CLEAR ENFORCE-Kommando

CLEAR ENFORCE

Ein Teil der Gerätenamen kann verkürzt angegeben werden.

Beispiele:

```
DS> SET ENFORCE
DS> ATTACH PP06 HUB DB0
?? ERROR IN CONTROLLER PART OF NAME, FORMAT SHOULD BE "DBAN"
DEVICE NAME
CTRL/C
DS> CLEAR ENFORCE
DS> ATTACH PP06 HUB DB0
DS> SHOW DEVICE
_DB0 PP06 HUB 00000000
```

SHOW CALLS-Kommando

SHOW CALLS

Es werden Frame, PC, AP und PSW der abgearbeiteten Call-Befehle ausgeschrieben.

17.2.2. Scripting

Die Möglichkeit des Scripting erlaubt dem Operator, vorher definierte Kommandofolgen automatisch abzuarbeiten. Die Supervisorkommandos, die normalerweise über das Operatorterminal eingegeben werden, werden statt dessen aus einer Text-Datei genommen.

SCRIPTING-Kommando

@<file-spec>

Dieses Kommando veranlaßt den Supervisor zur Abarbeitung der Kommandos, die in der spezifizierten Datei enthalten sind. Die Datei wird mit dem Text-Editor

erarbeitet. Zu Beginn jeder Zeile ist DS> (nur mit Großbuchstaben und mit Leerzeichen nach spitzer Klammer!) zu schreiben. Danach wird die Datei in die gleiche Directory kopiert, auf der der Supervisor steht. Steht diese Datei nicht in der gleichen Directory wie der Supervisor, ist das Gerät und die Directory im Kommando zu spezifizieren oder vorher mit dem SET LOAD-Kommando einzugeben. Sind in der Dateispezifikation Typ und Versionsnummer nicht enthalten, nimmt der Supervisor den Standard-Dateityp .COM und die höchste Versionsnummer.

Beispiel für Kommando-Datei:

```
DS> ATTACH DWA40 HUB DW0 3 4
DS> ATTACH AMF18A DW0 TXA 760340 314 5 377 9600 INTERNAL YES
DS> ATTACH TG911 TXA TXA0
DS> SELECT TXA0
DS> RUN PVTAA/PASS:3
```

Beispiel für Ausführung der Kommando-Datei:

```
$ COPY CMD.COM DBA0:[SYS0.SYSMAINT] !Kopieren der Kommando-Datei
$ RUN PSSAA                         !Start des Supervisors
    .
    .
    .
DS> @DBA0:[SYS0.SYSMAINT]CMD        !Abarbeitung der Kommando-Datei
```

Bemerkung:

COPY und RUN sind SVP-Kommandos

Die Diagnoseprogramme benötigen mit Ausnahme außergewöhnlicher Umstände keine Informationen vom Operator. Ausnahmen sind zum Beispiel dazwischengeschobene Tests und Vergleichsfehler von Programmen, die auf den Plattenspeicher zugreifen. Eingaben auf Bedienanforderungen sollten in diesen Fällen vom Operator und nicht vom Script erfolgen. Aus diesem Grund enthalten die Script-Dateien nur Supervisor-Kommandos.

Abarbeitung der Indirekt-Kommandos:

Der Supervisor arbeitet ein Indirekt-Kommando grob wie folgt ab:

1. Der Supervisor bricht das laufende Programm, wenn notwendig, ab.
2. Der Supervisor liest die gesamten Script-Datei auf einmal in einen Puffer.
3. Der Supervisor stellt einen Zeiger auf die erste Zeile der Script- Datei.
4. Der Supervisor setzt ein Flag zur Kennzeichnung, daß das nächste Kommando von einer Script-Datei stammt.
5. Mit Abarbeitung der Kommandos der Script-Datei werden auf dem Operatorterminal die Eingabeanforderung DS> und der Kommando-Text ausgegeben, sofern das Flag VERIFY gesetzt ist.
6. Nach Abarbeitung aller Kommandos der Script-Datei meldet sich der Supervisor mit DS>.

Pufferzuweisung und Script-Verschachtelung:

Der Supervisor reserviert dynamisch Speicherbereiche für Script-Text, Steuer- und Positionsinformationen. Jeder Script-Descriptor ist verbunden mit dem vorhergehenden Script-Descriptor. Dies ermöglicht das satzweise Abspeichern der Scripts. Der verfügbare Speicher des Rechnersystems begrenzt die Anzahl der

möglichen satzweise abzuspeichernden Scripts. Wird die Speicherkapazität überschritten, schreibt der Supervisor

UNKNOWN ERROR PC = 00000124 aus.

Die satzweise abgespeicherten Scripts können mit dem Kommando @<file-spec> von einer Script-Datei aufgerufen werden. Der Supervisor arbeitet die Kommandos der zweiten Script-Datei bis zum Ende ab. Danach wird die Abarbeitung der Kommandos von der ersten Script-Datei fortgesetzt. Nach Abarbeitung aller Kommandos erfolgt die Steuerung wieder über das Operatorterminal.

Unterbrechung der Script-Datei:

Die Abarbeitung einer Script-Datei kann durch Eingabe von CTRL/C über das Terminal unterbrochen werden. Der Supervisor bricht den Script ab und stoppt ein eventuell laufendes Programm. Danach können über das Terminal Kommandos eingegeben werden. Soll die Abarbeitung des Scripts durch Eingabe des Kommandos CONTINUE später fortgesetzt werden, ist die Kommandoauswahl begrenzt.
Diese Kommandos können eingegeben werden, wenn eine Fortsetzung der Abarbeitung der Script-Datei vorgesehen ist:

SET	SHOW	CLEAR	SUMMARY
EXAMINE	NEXT	DEPOSIT	CONTINUE

Folgende Kommandos führen zum Abbruch des Scripts, wobei die Steuerung zum Kommandozeilen-Interpreter des Supervisors übergeht:

ATTACH	START	SELECT	RUN
DESELECT	ABORT		

Kommando-Datei-Format:

Die Kommando-Datei muß aus ASCII-Zeichen bestehen, erzeugt durch RMS in einer ODS-1 oder ODS-2 Disk-Datei-Struktur. Die Datei ist zeilenorientiert mit max. 72 Zeichen pro Zeile. Eine Kommando-Datei kann erzeugt werden mit irgend einem Editor oder mit dem SVP-Kommando CREATE. Der Supervisor behandelt alle Aufzeichnungen als Supervisor-Kommandos. Alle gültigen Supervisor-Kommandos sind im Script zugelassen.

17.2.3. Abarbeitungssteuerfunktionen

Die Abarbeitungssteuerfunktionen gestatten, die Eigenschaften der Diagnoseprogramme und des Supervisors zu variieren. Diese Funktionen werden über Kommando-Flags und Ereignis-Flags realisiert. Die Kommando-Flags werden genutzt zur Steuerung von Fehlermitteilungen, Signalen und Stop bzw. zur zyklischen Abarbeitung der Programme.

SET FLAGS-Kommando

SET [FLAGS] <arg-list>

Dieses Kommando führt zum Setzen von Abarbeitungssteuer-Flags, die in der Argumentenliste spezifiziert sind. Die Argumentenliste besteht aus einer Kette von Mnemonics aus der folgenden Tabelle, die durch Kommas getrennt werden.

HALT Halt bei Fehler. Wenn das Programm einen Fehler festgestellt hat und dieses Flag ist gesetzt, tritt der Supervisor nach Ausgabe aller Fehlermitteilungen, die in Beziehung zu diesem Fehler stehen, in den Kommandowarte-Status ein. Das Programm kann fortgesetzt, erneut gestartet oder abgebrochen werden. Dieses Flag hat Vorrang vor dem LOOP-Flag.

BINARY Ist dieses Flag gesetzt, wird vor jede Ausschrift (einschließlich der Eingabeanforderungen und der Diagnoseprogrammausgaben) ein Byte ausgegeben, daß die Ausschrift klassifiziert (z.B. Kommandofehler, Programmstart, Hardwarefehler usw.). Der Inhalt des Bytes ist in den Programmen zu definieren.
Der Grundzustand des Flags ist zurückgesetzt und es wird durch die Kommandos SET FLAGS ALL oder CLEAR FLAGS ALL beeinflußt.

LOOP Schleife bei Fehler. Wenn dieses Flag gesetzt ist, wird ein vorher bestimmter Programmabschnitt eines Tests oder Subtests in einer Schleife abgearbeitet, so daß man den Fehler feststellen kann. Wenn das IE1-Flag gesetzt ist, werden die Fehlermitteilungen unterdrückt. Beachte, daß damit aber keine Mitteilungen des Programms oder des Supervisors verboten werden. Die Programmschleife wird solange abgearbeitet, bis der Operator die Steuerung mit CTRL/C an den Supervisor übergibt. Danach kann das Programm fortgesetzt, das Flag gelöscht und das Programm fortgesetzt oder abgebrochen werden.

BELL Signal bei Fehler. Bei gesetztem Flag sendet der Supervisor jedesmal ein Signal an den Operator, wenn das Programm einen Fehler feststellt.

IE1 Verbieten von Fehlermitteilungen der Ebene 1. Wenn dieses Flag gesetzt ist, werden alle Fehlermitteilungen unterdrückt mit Ausnahme derjenigen, die durch den Supervisor und das Programm erzeugt werden.

IE2 Verbieten von Fehlermitteilungen der Ebene 2. Wenn dieses Flag gesetzt ist, werden die Basis- und die erweiterten Informationen, die den Fehler betreffen, unterdrückt. Nur der Kopf der Fehlermitteilung (die ersten drei Zeilen) werden bei jedem Fehler ausgegeben.

IE3 Verbieten von Fehlermitteilungen der Ebene 3. Wenn dieses Flag gesetzt ist, werden die erweiterten Fehlerinformationen unterdrückt. Der Kopf und die Basisinformationen werden für jeden Fehler ausgegeben.

IES Verbieten Summary-Report. Wenn dieses Flag gesetzt ist, werden die Statistical Reports unterdrückt.

QUICK Schnelltest. Das gesetzte Flag zeigt dem Programm an, daß der Operator einen Schnelltestmodus wünscht. Die Interpretation dieses Flags ist programmabhängig.

VERIFY Ist das Flag gesetzt, werden die von der Script-Datei gelesenen Kommandos einschließlich der Eingabeanforderungen auf das Operatorterminal ausgegeben. Fehler- und sonstige Mitteilungen sowie Kommandos, die von einer SVP-Script-Datei gelesen werden, sind davon nicht betroffen.

TRACE Mitteilung über die Ausführung jedes Tests. Das gesetzte Flag veranlaßt den Supervisor, die Ausführung jedes einzelnen Tests innerhalb des Programms zu protokollieren, obwohl der Supervisor die Steuerung an den Test abgibt.

OPERATOR Operator anwesend. Das gesetzte Flag zeigt dem Supervisor an, daß eine interaktive Arbeit mit dem Operator möglich ist. Wenn das Flag zurückgesetzt ist, leitet der Supervisor geeignete Maßnahmen ein, um die Weiterarbeit des Tests ohne Operator sicherzustellen.

PROMPT Anzeige langer Dialog. Das gesetzte Flag zeigt dem Supervisor an, daß der Operator die Limits und Standards von allen Werten, die durch das Programm ausgegeben werden, angezeigt haben möchte.

SEARCH Einschalten Fehlersuchmodus. Wenn dieses Flag gesetzt ist, wird automatisch eine Modifikation des Testlimits vorgenommen. Wurde durch einen Test ein Fehler mitgeteilt, wird das nächste Testlimit auf den Wert des vorhergehenden Tests gesetzt. Mit einem unendlichen Zyklenzähler (ist Standard, wenn das SEARCH-Flag gesetzt ist) werden die Testzyklen eventuell soweit reduziert, daß nur noch die Tests einschließlich dem Test mit der kleinsten Nummer, der keinen Fehler brachte, abgearbeitet werden.

ALL Alle Flags dieser Liste werden gesetzt mit Ausnahme der Flags SEARCH und BINARY, die nicht verändert werden (weder gesetzt noch zurückgesetzt).

CLEAR FLAGS-Kommando

CLEAR [FLAGS] <arg-list>

Dieses Kommando führt zum Löschen der in der Argumentenliste spezifizierten Flags. Alle anderen Flags werden nicht beeinflußt. Die Argumentenliste besteht aus einer Kette von Mnemonics, die durch Komma getrennt werden. Argumente siehe SET FLAGS-Kommando.

SET FLAGS DEFAULT-Kommandos

SET FLAGS DEFAULT

Dieses Kommando setzt die Flags OPERATOR und PROMPT. Alle anderen Flags werden gelöscht.

SHOW FLAGS-Kommando

SHOW FLAGS

Dieses Kommando zeigt alle Abarbeitungssteuerflags und deren aktuellen Zustand. Die Flags werden in zwei Listen ausgegeben; eine Liste mit den gesetzten und eine Liste mit den gelöschten Flags.

Beispiel:

```
DS> SET FLAGS TRACE                          !Setzen TRACE-Flags
DS> CLEAR FLAGS QUICK                        !Löschen QUICK-Flag
DS> SHOW FLAGS
CONTROL FLAGS SET: PROMPT, OPER, TRACE
CONTROL FLAGS CLEAR: SEARCH, VERIFY, QUICK
                     IES, IE3, IE2, IE1,
                     BELL, LOOP, BINARY, HALT
```

SET EVENT FLAGS-Kommando

SET EVENT [FLAGS] <arg-list> !ALL

Dieses Kommando führt zum Setzen der Ereignis-Flags, die in der Argumentenliste spezifiziert wurden. Die anderen Ereignis-Flags werden nicht beeinflußt. Die Argumentenliste beinhaltet die Nummern der Flags im Bereich 1 - 23, die durch Kommas getrennt werden. Mit ALL werden alle Flags gesetzt.
Die Ereignis-Flags sind Statusbits, die durch das SVP und den Supervisor aufgestellt werden. Die Diagnoseprogramme können die Ereignis-Flags zur Veränderung von Mitteilungsfunktionen, einschließlich der Kommunikation mit dem Operator, nutzen.

Das SVP spezifiziert die Funtionen der ersten zwei Ereignis-Flags.

Ereignis-Flag 1 erlaubt (wenn gesetzt) oder verbietet (wenn gelöscht) Error Logging unter SVP, Standard ist gelöscht.
Ereignis-Flag 2 erlaubt (wenn gesetzt) oder verbietet (wenn gelöscht) Wiederholungen im Fehlerfall unter SVP. Standard ist gelöscht.

Die anderen möglichen Ereignis-Flags (3 - 23) sind nicht speziell definiert. Die Diagnoseprogramme können die Flags für die interaktive Arbeit mit dem Operator nutzen, wobei deren Funktion in der Programmdokumentation aufzuführen ist. Zum Beispiel kann man ein Flag nutzen, um besondere Datenmuster zu spezifizieren.

CLEAR EVENT FLAGS-Kommando

CLEAR EVENT [FLAGS] <arg-list> !ALL

Dieses Kommando löscht die in der Argumentenliste spezifizierten Ereignis-Flags. Die anderen Flags werden nicht beeinflußt. Die Argumentenliste enthält die Flag-Nummern im Bereich 1 - 23, getrennt durch Komma. Mit ALL werden alle Flags gelöscht.

SHOW EVENT FLAGS-Kommando

SHOW EVENT [FLAGS]

Dieses Kommando veranlaßt den Supervisor, eine Liste aller gesetzten Ereignis-Flags auszugeben.

Beispiel:

```
DS> SET EVENT FLAGS 1, 9, 15
DS> CLEAR EVENT FLAGS 2, 6
DS> SHOW EVENT FLAGS
EVENT FLAGS SET: 15, 9, 1
DS>
```

SET QA FLAGS-Kommando

SET QAMULTIPLEPASS <n>

Mit "n" wird die Anzahl der Durchläufe im Multiple Pass Check festgelegt.

Standardwert ist 10, d.h. es werden 10mal nacheinander alle Tests abgearbeitet.

SET QATESTLOOPS <n>

Mit "n" wird festgelegt, wievielmal jeder Test abgearbeitet wird. Standardwert ist 100, d.h., es wird 100mal ein Test abgearbeitet, ehe zum nächsten Test übergegangen wird.

SET QASUBTESTLOOPS <n>

Mit "n" wird festgelegt, wievielmal jeder Subtest abgearbeitet wird. Wenn z.B. ein Diagnoseprogramm zwei Tests, Test 1 mit drei Subtests und Test 2 mit zwei Subtests, enthält und QASUBTESTLOOPS ist auf drei gesetzt, dann werden die Tests in folgender Reihenfolge ausgeführt:

```
1.1, 1.1, 1.1,
1.1, 1.2, 1.2, 1.2,
1.1, 1.2, 1.3, 1.3, 1.3,
1.1, 1.2, 1.3, 2.1, 2.1, 2.1,
2.1, 2.2, 2.2, 2.2
2.1, 2.2
```

Der Standardwert ist 100.

SET QADEFAULTS

Es werden den eventuell geänderten Flags ihre Standardwerte zugeordnet. Diese sind:

QAMULTIPLEPASS	10
QATESTLOOPS	100
QASUBTESTLOOPS	100

SHOW QA FLAGS-Kommando

SHOW QAMULTIPLEPASS
SHOW QATESTLOOPS
SHOW QASUBTESTLOOPS

Mit diesen Kommandos können die Einstellungen der Flags abgefragt werden.

SHOW QADEFAULTS

Mit diesem Kommando werden die Standardwerte (nicht die evtl. veränderten) der Flags angezeigt.

17.2.4. Debug- und Dienstprogramm-Kommandos

Diese Gruppe von Kommandos gibt dem Operator die Möglichkeit, Fehler einzukreisen und den Code der Diagnoseprogramme zu ändern. Der Supervisor gestattet das gleichzeitige Setzen von bis zu 15 Unterbrechungspunkten innerhalb des Programms. Der Operator kann auch das Image im Hauptspeicher prüfen und modifizieren.

SET BASE-Kommando

SET BASE <address>

Dieses Kommando lädt die spezifizierte Adresse in ein Softwareregister. Diese Adresse wird dann genutzt als Basisadresse für die Adresse, die in den Kommandos SET BREAKPOINT, CLEAR BREAKPOINT und DEPOSIT eingegeben werden. Beide werden addiert. Das SET BASE-Kommando ist nützlich bei Vergleichen mit den Codes in den Diagnoseprogrammlisten. Die Basis sollte die Basisadresse (siehe Program Memory Allocation Map) der Programmsektion sein, auf die Bezug genommen wird. Dann kann der PC-Inhalt der Liste direkt für den zu vergleichenden Speicherplatz in der Programmsektion genutzt werden. Bei der Eingabe der Adresse sind die eingestellten Standard-Qualifiers zu beachten.

Beispiel:

```
DS> SET BASE E00          !Setzen der Basisadresse an den Anfang der PSECT
                          !der auszuführenden Routine
DS>
```

Bemerkung:

Virtuelle Adresse = physische Adresse (Normalfall), wenn die Speicherverwaltung ausgeschaltet ist.

SHOW BASE-Kommando

SHOW BASE

Mit diesem Kommando kann der Wert der laufenden Basisadresse angezeigt werden.

Beispiel:

```
DS> SHOW BASE
THE CURRENT BASE ADDRESS IS 00000E00(X).
DS>
```

SET BREAKPOINT-Kommando

SET BREAKPOINT <address>

Dieses Kommando veranlaßt die Übergabe der Steuerung an den Supervisor, wenn der Befehlszähler die Adresse erreicht hat, die vorher durch dieses Kommando eingegeben wurde. Gleichzeitig können maximal 15 Unterbrechungspunkte im Diagnoseprogramm gesetzt werden. Bei der Eingabe der Adresse sind die eingestellten Standard-Qualifiers zu beachten.

Beispiel:

```
DS> SET BREAKPOINT 30     !Setzen eines Unterbrechungspunktes mit einem
                          !Offset von 30 zur Basisadresse
DS>
```

CLEAR BREAKPOINT-Kommando

CLEAR BREAKPOINT <address> !ALL

Dieses Kommando löscht den vorher gesetzten Unterbrechungspunkt im Speicher, der durch "address" spezifiziert ist. Existiert auf dieser spezifizierten Adresse kein Unterbrechungspunkt, wird keine Fehlermeldung ausgegeben. Mit ALL werden alle vorher definierten Unterbrechungspunkte gelöscht. Bei der Eingabe der Adresse sind die eingestellten Standard-Qualifiers zu beachten.

Beispiel:

```
DS> CLEAR BREAKPOINT 30     !Löschen des Unterbrechungspunktes mit einem
                            !Offset von 30 zur Basis-Adresse
DS>
```

SHOW BREAKPOINTS-Kommando

SHOW BREAKPOINTS

Dieses Kommando zeigt alle z.Z. gesetzten Unterbrechungspunkte an.

Beispiele:

```
DS> SHOW BREAKPOINTS                !Anzeige der z.Z gesetzten
CURRENT BREAKPOINTS: 00000E30(X)    !Unterbrechungspunkte
DS>
```

SET DEFAULT-Kommando

SET DEFAULT <arg-list>

Mit diesem Kommando werden die Standard-Datenlänge und/oder der Standard-Radix-Qualifier für die EXAMINE-, SET BREAKPOINT-, CLEAR BRAEKPOINT-, SET BASE- und DEPOSIT-Kommandos sowie die Qualifiers für die EXAMINE- und und DEPOSIT-Kommandos gesetzt. Werden beide angegeben, sind sie durch Komma zu trennen. Wird nur einer angegeben, wird der andere nicht beeinflußt. Nach dem Start des Supervisors sind HEX und LONG die Standard-Qualifiers.

Standard-Qualifiers können sein:

```
Datenlänge: Byte, Word, Long
Radix:      Hexadecimal, Decimal, Octal
```

Beispiel:

```
DS> SET DEFAULT BYTE, DECIMAL
DS>
```

SHOW DEFAULT-Kommando

SHOW DEFAULT

Mit diesem Kommando werden die eingestellten Standard-Qualifiers angezeigt.

Beispiel:

```
DS> SHOW DEFAULT
CURRENT RADIX: HEXADECIMAL.
CURRENT SIZE: LONGWORD.
DS>
```

EXAMINE-Kommando

EXAMINE [<qualifiers>] <address>

Das EXAMINE-Kommando zeigt die Speicheradresse und den Inhalt des Speichers in dem Format an, wie es durch die Qualifiers gefordert wird. Sind keine Qualifiers angegeben, werden die Standard-Qualifiers (siehe SET DEFAULT-Kommando) genommen. Die eingegebene Adresse wird nur durch die Standard-Qualifiers beeinflußt.

Mögliche Qualifiers:

/BYTE	Ausgabe eines Bytes
/WORD	Ausgabe eines Wortes
/LONGWORD	Ausgabe eines Langwortes
/HEXADECIMAL	Anzeige Adresse und Daten hexadezimal
/DECIMAL	Anzeige Adresse und Daten dezimal
/OCTAL	Anzeige Adresse und Daten oktal
/ASCII	Anzeige Daten in ASCII-Zeichen und hexadezimal und Adresse hexadezimal
/NEXT=<n>	Anzeige des Inhalts der spezifizierten und der nachfolgenden "n" Speicherplätze, "n" ist eine Dezimalzahl

Als Option ist die Spezifizierung der Adreßargumente mit %D für dezimal, %O für oktal oder %X für hexadezimal vor der Adresse möglich. Adreßausdrücke können auch sein R0 - R11, AP, FP, SP, PC, PSL. Für AP, FP, SP und PC ist auch die Eingabe R12 - R15 möglich. Inhalte interner Prozeßregister können mit dem Adreß-Argument Pxx angezeigt werden, wobei xx die Register-Nummer ist (Standard-Qualifier beachten).

Beispiel:

```
DS> EXAMINE 30              !Anzeige des Inhalts des Langwortes mit dem
                            !Offset 30 zur Basisadresse E00
    00000E30: D0513D01
DS>
```

DEPOSIT-Kommando

DEPOSIT [<qualifiers>] <address> <data>

Mit diesem Kommando können Daten auf einen Speicherplatz, spezifiziert durch "address", in der durch den Qualifier festgelegten Datenlänge und den Standard-Radix-Qualifier festgelegten Datenart geschrieben werden. Ist kein Qualifier spezifiziert, wird der Standard-Qualifier genommen. Die Spezifizierung der Adresse und der Daten mit %D für dezimal, %O für oktal oder %X für hexadezimal vor dem jeweiligen Parameter ist ebenfalls möglich. Inhalte interner Prozessorregister können mit dem Adreß-Argument Pxx geändert werden, wobei xx die Register-Nummer ist (Standard-Qualifiers beachten).

Mögliche Qualifier:

/BYTE	Beschreiben eines Bytes
/WORD	Beschreiben eines Wortes
/LONGWORD	Beschreiben eines Langwortes
/NEXT=<n>	Speichert die Daten auf den spezifizierten und die nachfolgenden "n" Speicherplätze. "n" ist eine Dezimalzahl.

Beispiele:

```
DS> DEPOSIT/W 30 0001 !Abspeichern 0001 in das Wort mit dem Offset
                      !30 zur Basisadresse
DS>
```

NEXT-Kommando

NEXT [<number-of-instructions>]

Dieses Kommando veranlaßt den Supervisor, einen Macrobefehl auszuführen. Ist die Anzahl (dezimal) spezifiziert, werden soviel Macrobefehle abgearbeitet wie angegeben. Der Supervisor zeigt nach Ausführung jedes Befehls den PC des nächsten Befehls und den Inhalt der nachfolgenden vier Bytes an. Dieses Kommando wird zur schrittweisen Abarbeitung von Programmabschnitten, wo Probleme auftreten, genutzt. Das Kommando NEXT ist erst anzuwenden, nachdem das Programm durch einen Unterbrechungspunkt gestoppt wurde.

Beispiel:

```
DS> NEXT                  !Ausführung des nächsten Befehles
00000E31: D0513D01
DS>
```

17.3. Diagnoseprogramme unter Supervisor

17.3.1. Starten des Supervisors im User-Modus

Die Abarbeitung von Diagnoseprogrammen im User-Modus erfordert im allgemeinen folgende SVP-Privilegien und Quoten (als Minimum):

Privilegien:

GRPNAM	LOG_IO	PSWAPM
ALLSPOOL	GROUP	TMPMBX
DETACH	PRMCEB	PHY_IO
DIAGNOSE	PRMMBX	

Quoten:

CLI:	DCL	FILLM:	100
ASTLM:	1000	WSQUO:	512
DIOLM:	1000	PRI:	4
WSDEF:	256	BYTLM:	65000
PRCLM:	100	TQELM:	1000

BIOLM: 1000 PGFLQUO: 40

Werden durch einzelne Diagnoseprogramme weitere Privilegien benötigt, ist dies in der jeweiligen Beschreibung ausgewiesen. Nach dem Einloggen in das SVP über Nutzername und Paßwort können die Privilegien und Quoten mit dem Kommando

SHOW PROCESS/PRIVILEGES/QUOTAS !ALL

angezeigt werden. Mit dem Kommando

SET PROCESS/PRIVILEGES=(privilege[,...])

können die entsprechenden Privilegien gesetzt werden, sofern diese durch den Systemoperator in den UAF eingetragen wurden. Mit dem Kommando

RUN <file-spec>

wird der Supervisor gestartet. Der Dateiname für den Supervisor ist PSSAA. Device und Directory können weggelassen werden, wenn vorher mit dem Kommando

SET DEFAULT <device>:[<directory>]

die Standard-Directory eingetragen wurde, in der der Supervisor steht. Nach der Meldung des Supervisors mit DS> wird mit den Supervisor-Kommandos weitergearbeitet.

Beispiele:

```
$ RUN DBA0:[SYS0.SYSMAINT]PSSAA


DS>

$ SET DEFAULT DBA0:[SYS0.SYSMAINT]

$ RUN PSSAA


DS>
```

17.3.2. Starten des Supervisors im Stand-alone-Modus

17.3.2.1. Rechner einschalten

- Einlegen der Diskette DS1 in das Diskettenlaufwerk des Konsolsystems
- Netz zuschalten
- Schalterstellung des Konsolpaneels:
  AUTO RESTART - OFF
  Schlüsselschalter - LOCAL

Der Bootlader wird von der Diskette geladen und gestartet, der wiederum das Programm CONSOL.SYS (Betriebssystem des Konsolrechners) von der Diskette lädt und startet. Der Konsolrechner befindet sich jetzt im Konsol-E/A-Modus und meldet sich mit der Eingabeanforderung >>>. In diesem Modus erfolgt die Kommunikation

mit Hilfe der Kommandos der Konsolkommandosprache.

17.3.2.2. Booten des Supervisors von der Systemplatte

Die Konsol-Diskette DS1 enthält die Boot-Kommando-Dateien. Jeweils einen für die entsprechende Laufwerksnummer (0 - 7), unterteilt in die Typen unterschiedlicher Plattenspeichersysteme. Weiterhin gibt es Eingabeanforderungs-Dateien (einen für jeden Speichertyp) zum Booten des Supervisors von einem Plattenspeicher, der kein Systemgerät ist, oder von einer anderen Directory als SYSMAINT. Diese Dateien fragen den Operator nach dem Namen der Boot-Datei (z.B. DIAGBOOT.EXE) und nach dem Namen der zu bootenden Datei (z.B. PSSAA.EXE). Wird eine dieser Dateien genutzt, muß vor der Eingabe des BOOT-Kommandos in das Register R3 die Nummer des Plattenspeichers eingetragen werden, auf dem der Supervisor steht (mit DEPOSIT-Kommando).
Das BOOT-Kommando hat den allgemeinen Aufbau Sgn.

S= Booten der Diagnoseprogramme
g= Plattenspeichertyp, wobei
B= PP05/PP06
R= PM80 bedeuten.
n= Gerätenummer (0 - 7) oder A bei Aufruf der Eingabeanforderungs-Datei

DEPOSIT- und BOOT-Kommando können eingegeben werden, wenn sich der Konsolrechner im Konsol-E/A-Modus befindet und sich mit >>> meldet. Aus dem Arbeitszustand des Rechners gelangt man in diesen Modus über folgende Schritte:

- Eingabe CTRL/P (Konsolrechner geht vom Programm-E/A-Modus in Konsol-E/A-Modus)
- Eingabe HALT (Rechner geht in Halt-Zustand)

Der weitere Ablauf ist:

- Einlegen der Konsoldiskette
- Auflegen des SVP-Plattenstapels mit dem Diagnosesystem
- Booten des Supervisors
- Supervisor meldet sich mit DS>
- Eingabe von Supervisor-Kommandos

Beispiele:

```
>>> BOOT SB0                !Booten des Supervisors von der PP05, Laufwerk 0
     CPU HALTED
     INIT SEQ DONE
     HALT INST EXECUTED
     HALTED ATT 20003552

     G 0000000E  00000200
     LOAD DONE, 00005E00 BYTES LOADED
RVS DIAGNOSTIC SOFTWARE
     .
     .
     .
DS>

>>> DEPOSIT R3 2
>>> BOOT SRA                !Booten des Supervisors von PM80, Laufwerk 2
                            !über Eingabeanforderungs-Datei
```

```
    CPU HALTED
    INIT SEQ DONE
    HALT INST EXECUTED
    HALTED AT 20003552
    G 0000000E  00000200
    LOAD DONE, 00005E00 BYTES LOADED
Bootfile: [SYS0.SYSMAINT]DIAGBOOT.EXE

Enter Name of file: [SYS0.SYSMAINT]PSSAA.EXE
RVS DIAGNOSTIC SOFTWARE
     .
     .
     .
DS>
```

17.3.2.3. Booten des Supervisors von der Diskette

Ist ein Booten des Supervisors vom Plattenstapel nicht möglich (z.B. wegen Hardware-Fehlern), kann dieser über das Konsoldiskettenlaufwerk gebootet werden. Dazu ist die Diskette DS8 in das Konsollaufwerk einzulegen und mit dem Kommando BOOT der Supervisor zu booten. Nach der Meldung des Supervisors mit DS> wird mit den Supervisor-Kommandos weitergearbeitet.
Eine ausführliche Beschreibung des Startens des Supervisors ist im Kapitel 1 enthalten.

17.3.2.4. Nutzung von Script-Dateien

Es ist sinnvoll, daß in der SYS0.SYSMAINT zwei Script-Dateien erstellt werden, die den Test der Gesamtkonfiguration erleichtern. Die Konfigurations-Sript-Datei CONFIG.COM enthält eine Summe von ATTACH-Kommandos, die die Hardware-Konfiguration für den Supervisor beschreiben. Die Testdatei SYSTEST.COM enthält den Aufruf der CONFIG-Datei, wählt die Geräte für den Test aus, stellt die Flags ein und ruft die entsprechenden Diagnoseprogramme auf. Die Datei SYSTEST läuft aber nur im Stand-alone-Modus. Die Script-Dateien werden durch @SYSTEST oder @CONFIG gestartet. Die Namen der Dateien sind [SYS0.SYSMAINT]CONFIG.COM und [SYS0.SYSMAINT]SYSTEST.COM. Die Script-Dateien können mit einem Editor geändert werden. Werden diese Script-Dateien nicht genutzt, sind die entsprechenden Kommandos einzeln einzugeben.

Beispiele:

```
DS> !CONFIGURATIONS-SCRIPT-DATEI
DS> !CONFIG.COM
DS> !
DS> !ZENTRALE VERARBEITUNGSEINHEIT, HAUPTSPEICHER
DS> ATT KAA40 HUB KA0 YES YES 1FFF 1
DS> ATT MSA40 HUB MS0 1 8
DS> !
DS> !MSBUS-ADAPTER
DS> ATT RHA40 HUB RH0 8 5
DS> !
DS> !PLATTENSPEICHER
DS> ATT PM80 RH0 DRA1
```

```
DS> ATT PP06 RH0 DBA4
DS> ATT PP06 RH0 DBA5
DS> !
DS> !SKRBUS-ADAPTER
DS> ATT DWA40 HUB DW0 3 4
DS> !
DS> !MULTIFUNKTIONS-KOMMUNIKATIONS-KONTROLLER
DS> ATT AMF18A DW0 TXA 760340 314 5 377 9600 INTERNAL YES
DS> ATT AMF18P DW0 LCA 760340 314 5 LP
DS> !
DS> !TERMINALS
DS> ATT TG911 TXA TXA0
DS> ATT TG911 TXA TXA1
DS> ATT TG911 TXA TXA2
DS> !
DS> !DRUCKER
DS> ATT DP27 LCA LCA0
```

```
DS> !SYSTEMTEST-SCRIPT-DATEI
DS> !SYSTEST.COM
DS> !
DS> @CONFIG
DS> !
DS> SELECT ALL
DS> !
DS> SET FLAGS QUICK
DS> RUN PVRAA              !Test Plattenspeicher
DS> RUN PVTAA              !Test Terminal
DS> RUN PVAAA              !Test Drucker
DS> CLEAR FLAGS QUICK
```

17.4. Übersicht über die Supervisorkommandos

		Seite
ABORT		17-11
ATTACH	<UUT-type> <link-name> <generic-device-name>	17-4
CLEAR BREAKPOINT	<address> !ALL	17-22
CLEAR ENFORCE		17-14
CLEAR EVENT [FLAGS]	<arg-list> !ALL 1 - 23	17-19
CLEAR [FLAGS]	<arg-list> ALL, BELL, BINARY, HALT, IES, IE1, IE2, IE3, LOOP, OPER, PROMPT, QUICK, SEARCH, TRACE, VERIFY	17-18
CONTINUE		17-10
CTRL/C		17-10
CTRL/Y		17-11
CTRL/Z		17-11
DEATTACH oder DEATTACH	<generic-device-name> /ADAPTER=<link-name> /ADAPTER=<link-name> <generic-device-name>	17-5
DEPOSIT	[<qualifiers>] <address> <data> /B, /W, /L, /NEXT=<n>	17-24
DESELECT	[<qualifiers>] <generic-device-name>[:]- [,<generic-device-name>[:]...] !ALL /ADAPTER=<name>	17-6
DIRECTORY	[<file-spec>] [<qualifiers>] /WIDE	17-3
EXAMINE	[<qualifiers>] <address> /B, /W, /L, /H, /D, /O, /A, /NEXT=<n>	17-23
EXIT		17-11
HELP		17-12
LOAD	<file-spec>	17-3
NEXT	[<number-of-instructions>]	17-24
RUN	<file-spec>[/SECTION:<section-name>]- [/TEST:<first>[:<last>!/SUBTEST:<num>]]- [/PASSES:<count>][<qualifiers>] /QA	17-9
SELECT	[<qualifiers>] <generic-device-name>[:]- [,<generic-device-name>[:]...] !ALL /ADAPTER=<name>, /NOALLOCATE	17-6
SET BASE	<address>	17-21
SET BREAKPOINT	<address>	17-21
SET DEFAULT	<arg-list> Byte, Word, Long, Hexadecimal, Decimal, Octal	17-22
SET ENFORCE		17-14
SET EVENT [FLAGS]	<arg-list> !ALL 1 - 23	17-19
SET [FLAGS]	<arg-list> ALL, BELL, BINARY, HALT, IES, IE1, IE2, IE3, LOOP, OPER, PROMPT, QUICK, SEARCH, TRACE, VERIFY	17-16
SET FLAGS DEFAULT		17-18
SET LOAD	[<device>:][<directory>]	17-2
SET MEMORY	<n>	17-12
SET MM OFF		17-12

SET MM ON		17-12
SET PAGE	<n>	17-13
SET QADEFAULTS		17-20
SET QAMULTIPLEPASS	<n>	17-20
SET QASUBTESTLOOPS	<n>	17-20
SET QATESTLOOPS	<n>	17-20
SET WIDTH	<n>	17-13
SHOW BASE		17-21
SHOW BREAKPOINTS		17-22
SHOW CALLS		17-14
SHOW DEFAULT		17-23
SHOW DEVICE	[<qualifiers>] <generic-device-name>[:]- [,<generic-device-name>[:]...] /ADAPTER=<name>, /BRIEF	17-6
SHOW EVENT [FLAGS]		17-19
SHOW FLAGS		17-18
SHOW LOAD		17-2
SHOW MEMORY	[<qualifiers>] /ALL, /BUFFER, /DATA_STRUCTURE, /MAP	17-13
SHOW MM		17-12
SHOW QADEFAULTS		17-20
SHOW QAMULTIPLEPASS		17-20
SHOW QASUBTESTLOOPS		17-20
SHOW QATESTLOOPS		17-20
SHOW SECTIONS		17-4
SHOW SELECTED	[<qualifiers>] /BRIEF	17-7
SHOW STATUS		17-10
SHOW TESTS		17-3
SHOW WIDTH		17-13
START	[/SECTION:<section-name>]- [/TEST:<first>[:<last>!/SUBTEST:<num>]]- [/PASSES:<count>][<qualifiers>] /QA	17-7
SUMMARY		17-10
@	<file-spec>	17-14

Anlage A Begriffserklärung

Abkürzungen der Rechnerkonfiguration

Mnemonik	verbale Beschreibung	phys. Name	log. Name	Chiffre
AMF18	Multifunktions-Kommunikations-Kontroller	AMF18	TXa	K 8081
BDE	Terminal	TG911	TXa	K 8911.80
CPU	Zentrale Verarbeitungseinheit	KAA40	KAn	K 2810
FPA	Gleitkommabeschleuniger	FPA40		K 2812
GHE	G- und H-Gleitkommazusatz	KEA40		K 2813
KSS	Konsolsubsystem	KCA40		K 2811
MBA	MSBUS-Adapter	RHA40	RHn	K 2816
MEM	Hauptspeicher 2 Mbyte	MSA40	MSn	K 3581.10
MEME	2 Mbyte-Hauptspeichererweiterung	MSA40		K 3581.11
RVS	Rechner mit virtuellem Speicher	RVS		
SBA	SKRBUS-Adapter	DWA40	DWn	K 2815
UCS	Nutzersteuerspeicher	KUA40		K 2814

BLP-Abkürzungen

Mnemonik	verbale Beschreibung	Baugruppe	BLP-Nr.
AFS1	AFS K5163, Anschlußsteuerung für Folienspeicher	KSS	2641
AFS2	AFS K5163, Anschlußsteuerung für Folienspeicher	KSS	2651
AFS3	AFS K5163, Anschlußsteuerung für Folienspeicher	KSS	2660
AIS02	AIS K8060.02, Anschlußsteuerung für serielles Interface	KSS	1760
AMFC0	Multifunktions-Kommunikationskontroller	SBE	9201
AMFP1	AMF-Prüfstecker (seriell)	AMF18	9125
AMFP2	AMF-Prüfstecker	AMF18	9126
AMFV1	AMFV-Grundplatte	AMF18	9120
AMFV2	AMFV-IFSS-Modul	AMF18	9121
AMFV3	AMFV-V.24-Modul	AMF18	9122
AMFVS	AMFV-Spannungszuführung	SBE	9202
BAA	KSS-BUS-Abschluß/Adapter	KSS	8070
CAM	Cache-Adressmatrix	CPU	8904
CDM	Cache-Datenmatrix	CPU	8905
CEH	Exception-Steuerung	CPU	8914
CIB	Konsolrechner-Interface	CPU	8929
CLK	Taktsteuerung	CPU	8916
DAP	Rechenwerk-Steuerung	CPU	8913
DBP	Rechenwerk-Anpassung	CPU	8909
DCP	Rechenwerk <07:00>	CPU	8912
DDP	Rechenwerk <15:08>	CPU	8911
DEP	Rechenwerk <31:16>	CPU	8910
FAD	Gleitkomma-Adder	FPA	8927
FCT	Gleitkomma-Steuerung	FPA	8928
FMN	Gleitkomma-Multiplizierer 1	FPA	8925
FML	Gleitkomma-Multiplizierer 2	FPA	8926
FNM	Gleitkomma-Normalisierer	FPA	8924
FP-LP	KSS-Frontpaneel	KSS	1664
ICL	Interruptsteuerung	CPU	8915
IDP	Befehlspuffer	CPU	8907
IRC	Befehlsdekodierung	CPU	8908
KSI1	Kurzschlußstecker für IFSS-Anschluß	KSS	2000
KSI3	Kurzschlußstecker für IFSS-Anschluß	KSS	2290
MAY	MEM-Matrix	MEM	8813
MBE	MSBUS-Exerciser	MBA	
MCN	MEM-Kontroller	MEM	8812
MCP	MBA-Steuerung	MBA	8504
MDP	MBA-Datenkanal	MBA	8503
MET	MBA-Tester	MBA	
MIR	MBA-Register	MBA	8502
MPC	MBA-Anschlußplatte	MBA	8506
MSB	MEM-Synchronbus-Interface	MEM	8811
MSI	MBA-Synchronbus-Interface	MBA	8501
OPS	OPS K3564.01, Dynamischer Operativspeicher	KSS	1730
PCS	fester Mikroprogrammspeicher	CPU	8922
RCPU1	CPU-Rückverdrahtung 1	CPU	8001
RCPU2	CPU-Rückverdrahtung 2	CPU	8002

Mnemonik	verbale Beschreibung	Baugruppe	BLP-Nr.
RKSS	KSS-Rückverdrahtung	KSS	8006
RMBA	MBA-Rückverdrahtung	MBA	8003
RMEM	MEM-Rückverdrahtung	MEM	8005
RPMA	PMA-Rückverdrahtung	PMA	8009
RSBA	SBA-Rückverdrahtung	SBA	8004
RSBE1	SBE-Rückverdrahtung 1 (4-fach)	SBE	8011
RSBE2	SBE-Rückverdrahtung 2 (9-fach)	SBE	8010
SAD	BLP-Adapter	PRÜF	8105
SADE	BLP-Adapter (erweitert)	PRÜF	8104
SAK	SKRBUS-Adapterkarte (K4281)	SBE	9102
SBH	Synchronbussteuerung <31:16>	CPU	8903
SBL	Synchronbussteuerung <15:00>	CPU	8902
SBR	SKRBUS-Brücke (K4283)	SBE	9101
SBV1	SKRBUS-Verbinder (K4284)	SBE	9105
SBV2	SKRBUS-Verbinder (K4284)	SBE	9106
SBX	SKRBUS-Exerciser	SBE	9107
SCP	System-Bedienfeld	SCH1	8071
SET	SKRBUS-Tester	SBE	9104
SFT	ISB-Terminator	ISB	8073
STR	SKRBUS-Terminator (K4282)	SBE	9103
TBM	Adress-Übersetzungspuffer	CPU	8906
TRS	Synchronbus-Terminator/SILO	CPU	8901
UAI	SBA-Adress/Interrupt	SBA	8704
UBT	SBA-Terminator	SBA	8705
UCB	SBA-Steuerung	SBA	8702
UMD	SBA-Register/Datenkanal	SBA	8703
UPC	SBA-Anschlußplatte	SBA	8706
USC	Mikroprogrammsteuerung	CPU	8923
USI	SBA-Synchronbus-Interface	SBA	8701
WCS	ladbarer Mikroprogrammspeicher	CPU	8920
ZVE1	ZVE K2662, Zentrale Verarbeitungseinheit K1620	KSS	1151
ZVE2	ZVE K2662, Zentrale Verarbeitungseinheit K1620	KSS	1140
ZVE3	ZVE K2662, Zentrale Verarbeitungseinheit K1620	KSS	1130

Registerabkürzungen

Mnemonik	verbale Beschreibung	Baugruppe	Position
ACCR	Gleitkommabeschleuniger-Maintenance-Register	FMH 8925, FML 8926	ID:16 PR:29
ACCS	Gleitkommabeschleuniger-Steuer-/Statusregister	FMH 8925, FML 8926	ID:17 PR:28
ASTLVL	AST-Ebenenregister		PR:13
BCR	MBA-Bytezähler-Register	MIR 8502	
BDP	Prüfdatenregister	UMD 8703	
BRVRn	Interrruptvektor-Register n (n = 4...7)	UAI 8704	
CAR	MBA-Kommando-/Adress-Register	MIR 8502	
CES	CPU-Fehlerstatusregister	ICL 8915	ID:0C
CNFGR	SBA-Konfigurations-Register	USI 8701	
CPER	Cache-Paritätsfehler-Register	SBL 8902	ID:1E
CR	MBA-Steuerregister	MIR 8502	
CSR	MBA-Konfigurations-/Statusregister	MIR 8502	
DCR	SBA-Diagnose-Register	USI 8701	
DPRn	Datenpfad-Register n (n = 0...15)	UMD 8703	
DQ	Lesen: D-Register Schreiben: Q-Register		
DR	MBA-Diagnose-Register	MIR 8502	
ESP	Executive-Stack-Pointer		ID:29 PR:01
FMER	SBA-Register für fehlerhaften Map-Eintrittspunkt	UMD 8703	
FUBAR	SBA-Register für fehlerhafte SKRBUS-Adresse	UAI 8704	
ICCS	Intervalzeitgeber-Steuer-/Statusregister	ICL 8915	ID:0A PR:18
ICR	Zählregister für Intervall-zeitgeber	CEH 8914,	ID:0B PR:1A ICL 8915,
IDCS	Konsolregister: ID-Steuer-/Statusregister		Adresse: '173030
IDDH	Konsolregister: ID-Daten <31:16>		Adresse: '173010
IDDL	Konsolregister: ID-Daten <15:00>		Adresse: '173006
IOBUF	Befehlspuffer	IDP 8907	ID:00
IPL	Interrupt-Prioritätsebenen-Register		PR:12
ISP	Interrupt-Stack-Pointer		ID:2C PR:04
KSP	Kernel-Stack-Pointer		ID:28 PR:00
MBE_ASR	MBE-Attentionsammel-Register	MBE	
MBE_CR1	MBE-Steuerregister 1	MBE	
MBE_CR2	MBE-Steuerregister 2	MBE	
MBE_DBR	MBE-Datenpuffer-Register	MBE	
MBE_DTR	MBE-Drivertyp-Register	MBE	
MBE_ER	MBE-Fehlerregister	MBE	
MBE_MR	MBE-Maintenance-Register		
MBE_SR	MBE-Fehlerstatusregister	MBE	
MBRK	Mikroprogramm-Unterbrechungspunkt	UCS 8923	ID:21 PR:3C
MCR	Konsolregister: Geräte-Steuerregister		Adresse: '173032
MCRA	Speicherkonfigurationsregister Adresse: %X2000 2000 bei TR1		

Mnemonik	verbale Beschreibung	Baugruppe	Position
MCRB	Speicherkonfigurationsregister Adresse: %X2000 2004 bei TR1		
MCRC	Speicherkonfigurationsregister Adresse: %X2000 2008 bei TR1		
MCRD	Speicherkonfigurationsregister Adresse: %X2000 200C bei TR1		
MCRE	Speicherkonfigurationsregister Adresse: %X2000 2010 bei TR1		
MCRF	Speicherkonfigurationsregister Adresse: %X2000 2014 bei TR1		
MCS	Konsolregister: Geräte-Steuer-/Statusregister		Adresse: '173034
MRn	Map-Register n (n = 0...495)	UMD 8703	
MSR	MBA-Ausgewähltes-Map-Register	MIR 8503	
NICR	Register für nächsten Interval	CEH 8914, ICL 8915	ID:09 PR:19
P0BR	P0-Basisregister		ID:24 PR:08
P0LR	P0-Längenregister		ID:3C PR:09
P1BR	P1-Basisregister		ID:25 PR:0A
P1LR	P1-Längenregister		ID:3D PR:0B
PCBB	Prozess-Steuerblock-Basisregister		ID:3A PR:10
PSL	Prozessorstatuslangwort	CEH 8914, ICL 8915	ID:0F
RXCS	Konsolregister: Konsol-Empfangs-Steuer-/Status-register	CIB 8929	Adresse: '173014
RXDB	Konsolregister: Konsol-Empfangs-Datenpuffer	CIB 8929	ID:05 PR:21
RXDBH	Konsolregister: Konsol-Empfangs-Datenpuffer <31:16>		Adresse: '173022
RXDBL	Konsolregister: Konsol-Empfangs-Datenpuffer <15:00>		Adresse: '173020
SBIER	ISB-Fehlerregister	SBL 8902	ID:19 PR:34
SBIFS	ISB-Fehlerstatusregister	SBH 8903	ID:1B PR:30
SBIMT	ISB-Maintenance-Register	SBL 8902, SBH 8903	ID:1D PR:33
SBIS	ISB-SILO	SBH 8903, TRS 8901	ID:18 PR:29
SBISC	ISB-SILO-Vergleichsregister	SBH 8903	ID:1C PR:32
SBITA	ISB-Time-Out-Adress-Register	SBL 8902, SBH 8903	ID:1A PR:35
SBR	System-Basisregister		ID:26 PR:0C
SCBB	System-Steuerblock-Basisregister		ID:3B PR:11
SID	Systemidentifikations-Register	CIB 8929	ID:03 PR:3E
SIRR	Software-Anforderungsregister		PR:14
SISR	Software-Interrupt-Sammel-Register	ICL 8915	ID:0E PR:15
SLR	Systemlängen-Register		ID:3E PR:0D
SPARE	Konsolregister		Adresse: '173004
SR	MBA-Statusregister	MIR 8502	
SSP	Supervisor-Stack-Pointer		ID:2A PR:02
T0			ID:30
T1			ID:31
T2			ID:32

Mnemonik	verbale Beschreibung	Baugruppe	Position
T3			ID:33
T4			ID:34
T5			ID:35
T6			ID:36
T7			ID:37
T8			ID:38
T9			ID:39
TBDR	Übertragungspuffer-Datenregister	TBM 8906, CAM 8904	ID:10 PR:24
TBR0	Übertragungspuffer-Register 0	TBM 8906	ID:12
TBR1	Übertragungspuffer-Register 1	TBM 8906	ID:13
TODR	Zählregister für Jahresuhr	IRC 8908	ID:01 PR:1B
TXCS	Konsolregister: Konsol-Sende-Steuer-/Status-register	CIB 8929	Adresse: '173026
TXDB	Konsolregister: Konsol-Sende-Datenpuffer	CIB 8929	ID;07 PR:23
TXDBH	Konsolregister: Konsol-Sende-Datenpuffer <31:16>		Adresse: '173026
TXDBL	Konsolregister: Konsol-Sende-Datenpuffer <15:00>		Adresse: '173024
UBACR	SBA-Steuerregister	UAI 8704	
UBASR	SBA-Statusregister	UAI 8704	
USP	User-Stack-Pointer		ID:2B PR:03
USTACK	Mikrostack	UCS 8923	ID:20
VBUS	Diagnosebus-Register		Adresse: '173036
VAR	MBA-Register für virt. Adresse	MIR 8502	
VECT	Vektorregister	CEH 8914	ID:0D
WCSA	WCS-Adress-Register	UCS 8923	ID:22 PR:2C
WCSD	WCS-Datenpuffer	WCS 8920	ID:23 PR:2D

Programmspezifische Fehlerausschriften

Mnemonik	verbale Beschreibung	Programm	Bemerkung
ABORT	Datenübertragungsabbruch	PVCAA	CR: <01>
AD	Adresse	LPROEM	KSS
ADINIT	Adapter-Initialisierung	PVCBA	UBACR: <00>
AF	Adressfehler	LPROEM	KSS
ATA	Attention aktiv	PVCAA	MBE_SR: <15>
ATAn	Attention-Signal vom Gerät n	PVCAA	MBE_ASR: <00+n>
ATTN	Attention-Signal vom MSBUS	PVCAA	SR: <16>
BA	Blockanfangsadresse bei Datenblockfehler	LPROEM	KSS
BC	Befehlskode eines fehlerhaften Befehles	LPROEM	KSS
BD	Bedienfehler	LPROEM	KSS
BE	Blockendeadresse bei Datenblockfehler	LPROEM	KSS
BF	Befehlsablauf fehlerhaft	LPROEM	KSS
BLKSCOM	Blocksendekommando zum ISB	PVCAA	DR: <28>
CPE	Steuerparitätsfehler	PVCAA	MBE_ER: <03>
CRD	korrigierte Lesedaten	PVCAA	SR: <29>
Cn	Statusregister (n = 1...8)	LPROEM	KSS
DB	Datenblockfehler	LPROEM	KSS
DLT	verspätete Datenübertragung	PVCAA	SR: <11>
DP	Datenfehler	LPROEM	KSS
DPE	Datenparitätsfehler	PVCAA	MBE_ER: <04>
DPR	Laufwerk vorhanden	PVCAA	MBE_SR: <08>
DRY	Laufwerk lesend	PVCAA	MBE_SR: <07>
DSn	ausgewähltes Gerät n	PVCAA	MBE_ASR: <8+n>
DTABT	Abbruch der Datenübertragung	PVCAA	SR: <12>
DTBUSY	Datenübertragung belegt	PVCAA	SR: <31>
DTE	Laufwerkzeitsteuerfehler	PVCAA	MBE_ER: <12>
DT_COMP	vollständige Datenübertragung	PVCAA	SR: <13>
DVA	Laufwerk verfügbar	PVCAA	MBE_CR1: <11>
EE	Geräteeingriff erforderlich	LPROEM	KSS
ERR	komplexer Fehler	PVCAA	MBE_SR: <14>
ERR_CONF	Fehlerbestätigung	PVCAA	SR: <03>
FAIL	MSBUS FAIL	PVCAA	DR: <20>
GF	Gerätefehler	LPROEM	KSS
GK	Gleitkommafehler	LPROEM	KSS
IE	Interrupt-Freigabe	PVCAA	CR: <02>
IF	Interruptfehler	LPROEM	KSS
ILF	illegale Funktion	PVCAA	MBE_ER: <20>
IMAPP	invertierte Map-Parität	PVCAA	DR: <29>
IMCPG	Generierung der invertierten Steuerparität	PVCAA	DR: <30>
IMDPG	Generierung der invertierten Datenparität	PVCAA	DR: <31>
IN	interner Fehler (erkannte Unterbrechung)	LPROEM	KSS
INIT	Initialisierung des MBA u. MSBUS	PVCAA	CR: <00>
INVMAP	ungültiges Map	PVCAA	SR: <04>
IS	Istwert	LPROEM	KSS
ISBPE	ISB-Paritätsfehler	PVCAA	CSR: <31>
ISR4	Interrupt-Service-Routine 4	PVCBA	

Mnemonik	verbale Beschreibung	Programm	Bemerkung
ISTIMEOUT	Interface-Folge-Time-Out	PVCAA	SR: <01>
IV	Interruptvektoradresse	LPROEM	KSS
IVMR	ungültiges Map-Register	PVCBA	UBASR: <04>
MAPPE	Map-Paritätsfehler	PVCAA	SR: <05>
MBDIBSEL	MSBUS-Dateneingabepuffer-Auswahl	PVCAA	DR: <23>
MBEEXC	MSBUS-Exception	PVCAA	SR: <07>
MCPE	MSBUS-Steuerparitätsfehler	PVCAA	SR: <17>
MCTOD	MSBUS CTOD	PVCAA	DR: <17>
MDIBn	ausgewähltes MSBUS-Dateneingabepuffer-Byte	PVCAA	DR: <00+n>
MDPE	MSBUS-Datenparitätsfehler	PVCAA	SR: <06>
MDSn	MSBUS-Laufwerkauswahlfeld	PVCAA	DR: <13+6>
MEXC	MSBUS-Exception	PVCAA	DR: <17>
MF	Maskierungsfehler	LPROEM	KSS
MMM	Maintenance-Modus	PVCAA	CR: <03>
MOL	Medium online	PVCAA	MBE_SR: <12>
MRSn	MSBUS-Registerauswahlfeld	PVCAA	DR: <08+n>
MRUN	MSBUS RUN	PVCAA	DR: <19>
MT	Mehrfachtransmitter	PVCAA	CSR: <27>
MWCLK	MSBUS WCLK	PVCAA	DR: <18>
MXF	vermisste Datenübertragung	PVCAA	SR: <08>
NED	Laufwerk existiert nicht	PVCAA	SR: <18>
NRCONF	keine Antwortbestätigung	PVCAA	SR: <30>
OPI	invollständige Datenübertragung	PVCCAA	MBE_ER: <13>
PC	Befehlszähler zum Fehlerzeitpunkt	LPROEM	KSS
PD	MSBUS-Adapterspannung "AUS"	PVCAA	CSR: <23>
PGE	Programmierfehler	PVCAA	SR: <19>
PS	Prozessorstatuswort zum Fehlerzeitpunkt	LPROEM	KSS
PU	MSBUS-Adapterspannung "EIN"	PVCAA	CSR: <22>
RCPA	Steuerparität wurde vom Kontroller empfangen	PVCAA	MBE_SR: <03>
RDPA	Datenparität wurde vom Kontroller empfangen	PVCAA	MBE_SR: <04>
RDS	Lesedatenersatz	PVCAA	SR: <02>
RDTIMEOUT	READ-Daten-Time-Out	PVCAA	SR: <00>
RFAIL	Empfang von MSBUS FAIL	PVCAA	MBE_ER: <04>
RHBCR	MBA-Bytezähler-Register	PVCAA	MIR 8502
RHCR	MBA-Steuerregister	PVCAA	MIR 8502
RHCSR	MBA-Konfigurations-/Statusregister	PVCAA	MIR 8502
RHDR	MBA-Diagnose-Register	PVCAA	MIR 8502
RHMAPR	MBA-Map-Register	PVCAA	MIR 8502
RHSR	MBA-Statusregister	PVCAA	MIR 8502
RHVAR	MBA-Register für virt. Adresse	PVCAA	MIR 8502
RMBEXC	Empfang von MSBUS-Exception	PVCAA	MBE_ER: <05>
RMR	Verweigerung der Registermodifizierung	PVCAA	MBE_ER: <02>
RRUN	RUN vom Kontroller empfangen	PVCAA	MBE_SR: <01>
RSn	ausgewähltes Register n	PVCAA	MBE_ASR: <11+n>
Rn	allgemeine Register (n = 0...5)	LPROEM	KSS
S/B	should be	PVCBA	

Mnemonik	verbale Beschreibung	Programm	Bemerkung
SA	Speicheradresse (18 Bit)	LPROEM	KSS
SC	Statusmeldung serieller Geräte	LPROEM	KSS
SF	Statusfehler	LPROEM	KSS
SIMATTN	Simulation des ATTN-Signales	PVCAA	DR: <24>
SIMEBL	Simulation des EBL-Signales	PVCAA	DR: <26>
SIMOCC	Simulation des OCC-Signales	PVCAA	DR: <25>
SIMSCLK	Simulation des SCLK-Signales	PVCAA	DR: <27>
SK	Speicherfehler bei Speicher mit Fehlerkorrektur	LPROEM	KSS
SL	Sollwert	LPROEM	KSS
SP	Speicherfehler	LPROEM	KSS
SP	Stapelzeiger zum Fehlerzeitpunkt	LPROEM	KSS
SV	Zustandsregister der Speicherverwaltung	LPROEM	KSS
SZ	ZE-Status	LPROEM	KSS
T	Testabbruchkennzeichen	LPROEM	KSS
UBIC	SKRBUS-Initialisierung	PVCBA	CNFGR: <10>
UBPDN	SKRBUS-Adapterspannung "AUS"	PVCBA	CNFGR: <17>

Anlage B Diskettenübersicht

Diskette DS0: "Konsolsubsystemtests" 1.57.700100.2/67

Datei-Name	Dok.-Nr.
LPROEM	1.57.700200.3
ZVZW20	1.57.700201.1
ZARI20	1.57.700202.8
ZSHI20	1.57.700203.6
ZBBI20	1.57.700204.4
ZBEF30	1.57.700205.2
ZADRAR	1.57.700206.0
ZSPE20	1.57.700207.7
ZTRAPI	1.57.700208.5
ZBUSCI	1.57.700209.3
AKSAIS	1.57.700210.8
GFLOPM	1.57.700211.6
DAUGEN1	1.57.700250.1
DAUGEN2	1.57.700251.8
DAUGEN3	1.57.700252.6
DAUGEN4	1.57.700253.4
DAUGEN5	1.57.700254.2
DAUGEN6	1.57.700255.0

Diskette DS1: "K 1840 Standardkonsole" 1.57.700101.0/67

CONSOL.SYS	1.57.700300.4
VMB.EXE	1.57.700351.0
CONSOL.HLP	1.57.700275.1
ABBREV.HLP	1.57.700276.8
ERROR.HLP	1.57.700277.6
BOOT.HLP	1.57.700278.4
CS1BOO.CMD	1.57.700402.1
CS1GEN	1.57.700403.8
DB0BOO.CMD	1.57.700404.6
DB0BOO.ILV	1.57.700405.4
DB0GEN	1.57.700406.2
DB0XDT	1.57.700407.0
DB1BOO.CMD	1.57.700408.7
DB1GEN	1.57.700409.5
DB2BOO.CMD	1.57.700410.1
DB2GEN	1.57.700411.8
DB3BOO.CMD	1.57.700412.6
DB3GEN	1.57.700413.4
DB4BOO.CMD	1.57.700414.2
DB4GEN	1.57.700415.0
DB5BOO.CMD	1.57.700416.7
DB5GEN	1.57.700417.5
DB6BOO.CMD	1.57.700418.3
DB6GEN	1.57.700419.1
DB7BOO.CMD	1.57.700420.6
DB7GEN	1.57.700421.4
DBABOO.CMD	1.57.700422.2
DBAGEN	1.57.700423.0
	Dok.-Nr.
DBAXDT	1.57.700424.7

	DBBBOO.CMD	1.57.700425.5
	DBBGEN	1.57.700426.3
	DBBXDT	1.57.700427.1
	SB0BOO.CMD	1.57.700428.8
	SB1BOO.CMD	1.57.700429.6
	SB2BOO.CMD	1.57.700430.2
	SB3BOO.CMD	1.57.700431.0
	SB4BOO.CMD	1.57.700432.7
	SB5BOO.CMD	1.57.700433.5
	SB6BOO.CMD	1.57.700434.3
	SB7BOO.CMD	1.57.700435.1
	SBABOO.CMD	1.57.700436.8
	SBBBOO.CMD	1.57.700457.7
	DR0BOO.CMD	1.57.700437.6
	DR1BOO.CMD	1.57.700438.4
	DR2BOO.CMD	1.57.700439.2
	DR3BOO.CMD	1.57.700440.7
	DR4BOO.CMD	1.57.700441.5
	DR5BOO.CMD	1.57.700442.3
	DR6BOO.CMD	1.57.700443.1
	DR7BOO.CMD	1.57.700444.8
	DRABOO.CMD	1.57.700459.3
	DRBBOO.CMD	1.57.700460.8
	SR0BOO.CMD	1.57.700445.6
	SR1BOO.CMD	1.57.700446.4
	SR2BOO.CMD	1.57.700447.2
	SR3BOO.CMD	1.57.700448.0
	SR4BOO.CMD	1.57.700449.7
	SR5BOO.CMD	1.57.700450.3
	SR6BOO.CMD	1.57.700451.1
	SR7BOO.CMD	1.57.700452.8
	SRABOO.CMD	1.57.700453.6
	SRBBOO.CMD	1.57.700458.5
	CRASH	1.57.700454.4
	RESTAR.CMD	1.57.700455.2
	RESTAR.ILV	1.57.700456.0
	WCS124.PAT	1.57.700350.2
	DSCOPY.COM	1.57.700354.3
Diskette	DS2: "K 1840 MIKRODIA #1"	1.57.700102.7/67
	MICMON.SYS	1.57.700301.2
	PSKAC.TSK	1.57.700302.0
	PSKAD.TSK	1.57.700303.7
	PSKAE.TSK	1.57.700304.5
	PSKAF.TSK	1.57.700305.3
	PSKAG.TSK	1.57.700306.1
	PSKAK.TSK	1.57.700307.8
	PSKAH.TSK	1.57.700308.6
	PSKAL.TSK	1.57.700309.4

Diskette	DS3: "K 1840 MIKRODIA #2"	1.57.700103.5/67
		Dok.-Nr.
	MICMON.SYS	1.57.700301.2
	PSKAE.TSK	1.57.700304.5
	PSKAF.TSK	1.57.700305.3
	PSKAG.TSK	1.57.700306.1
	PSKAK.TSK	1.57.700307.8
	PSKAR.TSK	1.57.700310.0
	PSKAS.TSK	1.57.700311.7
Diskette	DS5: "BEFEH #1"	1.57.700105.1/67
	PVKAA.EXE	1.57.700001.8
	PVKAS.EXE	1.57.700002.6
	PVKAB.EXE	1.57.700003.4
	PVKAA.HLP	1.57.700501.4
	PVKAS.HLP	1.57.700502.2
	PVKAB.HLP	1.57.700503.0
Diskette	DS6: "BEFEH #2"	1.57.700106.8/67
	PVKAC.EXE	1.57.700004.2
	PVKAD.EXE	1.57.700005.0
	PVKAE.EXE	1.57.700006.7
	PVKAC.HLP	1.57.700504.7
	PVKAD.HLP	1.57.700505.5
	PVKAE.HLP	1.57.700506.3
Diskette	DS7: "BUS + AUTOKONFIG DIAG"	1.57.700107.6/67
	PVCAA.EXE	1.57.700009.1
	PVCBA.EXE	1.57.700010.6
	PVSBA.EXE	1.57.700008.3
	PVCAA.HLP	1.57.700509.6
	PVCBA.HLP	1.57.700510.2
	PVSBA.HLP	1.57.700508.8
Diskette	DS8: "K 1840 DIAG SUPER"	1.57.700108.4/67
	PSSAA.EXE	1.57.700000.1
	PVSAA.HLP	1.57.700500.6
	DEFBOO.COM	1.57.700400.5
	CONFIG.COM	1.57.700353.5
Diskette	DS9: "K 1840 DISK DIAG #1"	1.57.700109.2/67
	PVRBA.EXE	1.57.700604.8
	PVRBA.HLP	1.57.700754.7
	PVRCA.EXE	1.57.700605.6
	PVRCA.HLP	1.57.700755.5
Diskette	DS10: "K 1840 DISK DIAG #2"	1.57.700110.7/67
	PVRDA.EXE	1.57.700606.4
	PVRDA.HLP	1.57.700756.3

		Dok.-Nr.
	PVRGA.EXE	1.57.700607.2
	PVRGA.HLP	1.57.700757.1
	PVRGB.EXE	1.57.700608.0
	PVRGB.HLP	1.57.700758.8
Diskette	DS11: "K 1840 DISK DIAG #3"	1.57.700111.5/67
	PVQDB.EXE	1.57.700700.8
	PVQDR.EXE	1.57.700701.6
	PVRAA.EXE	1.57.700601.5
	PVRAA.HLP	1.57.700751.4
	PVRAC.EXE	1.57.700602.3
	PVRAC.HLP	1.57.700752.2
	PVRAD.EXE	1.57.700603.1
	PVRAD.HLP	1.57.700753.0
Diskette	DS12: "AMF18 K 8081 DIAG"	1.57.700112.3/67
	PVDLB.EXE	1.57.700802.5
	PVDLB.HLP	1.57.700952.4
	PVDLC.EXE	1.57.700803.3
	PVDLC.HLP	1.57.700953.2
	PVDLD.EXE	1.57.700804.1
	PVDLD.HLP	1.57.700954.0
Diskette	DS13: "AR16W K 6081 DIAG"	1.57.700113.1/67
	PVDRB.EXE	1.57.700809.0
	PVDRB.HLP	1.57.700959.8
	PVDRE.EXE	1.57.700810.5
	PVDRE.HLP	1.57.700960.4
Diskette	DS14: "RONAS K 8681 DIAG"	1.57.700114.8/67
	PVDWA.EXE	1.57.700806.6
	PVDWA.HLP	1.57.700956.5
Diskette	DS15: "K 1840 TAPE DIAG"	1.57.700115.6/67
	PVMZA.EXE	1.57.700613.6
	PVMZA.HLP	1.57.700763.5
	PVQTE.EXE	1.57.700702.4
	PVMZB.EXE	1.57.700614.4
	PVMZB.HLP	1.57.700764.3
	PVMAA.EXE	1.57.700615.2
	PVMAA.HLP	1.57.700765.1
	PVQTF.EXE	1.57.700703.2

Diagnoseprogramme der Ebene 2R
und weitere Dateien der Directory SYSMAINT

Datei-Name	Dok.-Nr.
DIAGBOOT.EXE	1.57.700352.7
PVKAM.EXE	1.57.700007.5

PVKAM.HLP	1.57.700507.1
PVXBA.EXE	1.57.700011.4
PVXBA.HLP	1.57.700511.0
PVXBB.EXE	1.57.700012.2
PVXBB.HLP	1.57.700512.7

Der aktuelle Stand der Diskettenübersicht und die Disketteninhalte sind in der jeweiligen Ausgabe des Diagnoseprogrammpaketes in der Datei AAPNX.LIS der Diagnose-Directory SYSMAINT untergebracht.